Neural network techniques for position and scale invariant image classification

Thesis

UNRESTRICTED

Neural Network Techniques for Position and Scale Invariant Image Classification

By

Catherine Alison Grimes BSc MSc

A thesis submitted for the degree of

DOCTOR OF PHILOSOPHY

of the Open University

March 1998

Department of Design and Innovation

The Open University

Walton Hall

Milton Keynes

ABSTRACT

This research is concerned with the application of neural network techniques to the problems of classifying images in a manner that is invariant to changes in position and scale. In addition to the goal of invariant classification, the network has to classify the objects in a hierarchical manner, in which complex features are constructed from simpler features, and use unsupervised learning. The resultant hierarchical structure should be able to classify the image by having an internal representation that models the structure of the image.

After finding existing neural network techniques unsuitable, a new type of neural network was developed that differed from the conventional multi-layer perceptron type of architecture. This network was constructed from neurons that were grouped into feature detectors. These neurons were taught in an unsupervised manner that used a technique based on Kohonen learning. A number of novel techniques were developed to improve the learning and classification performance of the network.

The network was able to retain the spatial relationship of the classified features; this inherent property resulted in the capability for position and scale invariant classification. As a consequence, an additional invariance filter was not required. In addition to achieving the invariance property, the developed techniques enabled multiple objects in an image to be classified.

When the network had learned the spatial relationships between the lower level features, names could be assigned to the identified features. As part of the classification process, the

system was able to identify the positions of the classified features in all layers of the network.

A software model of an artificial retina was used to test the grey scale classification performance of the network and to assess the response of the retina to changes in brightness.

Like the Neocognitron, the resulting network was developed solely for image classification. Although the Neocognitron is not designed for scale or position invariance, it was chosen for comparison purposes because it has structural similarities and the ability to accommodate slight changes in the image.

This type of network could be used as the basis for a 2D-scene analysis neural network, in which the inherent parallelism of the neural network would provide simultaneous classification of the objects in the image.

ACKNOWLEDGEMENTS

The author wishes to thank Dr. Anthony Lucas-Smith, Dr. Phillip Picton and Dr. David Elliman for their encouragement and support throughout the period of this research.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1.0 INTRODUCTION

Vision can be described as the process of discovering from images what is present in the world, and where it is [Marr 1982]. The field of vision research is enormous, encompassing the skills of psychologists, neurobiologists, and computer scientists. This research considers one small aspect of vision, namely translation and scale invariance.

When we look for the first time at an easily recognisable object, for example a new model of car, provided that we know what a car looks like, we can instantly recognise the object, and classify it as a car, even though we have never seen it before. This is still true when viewing the car from a different position, in which the image of the car's size, orientation, etc. are different. One could go on listing further possible differences in the images of the car. The visual processing in the brain is truly remarkable.

Further to the recognition of the object, the brain can even perform structural decomposition. The position of parts of the car can be estimated, and their relationship to each other determined. For example, the front doors being just behind the front wheels. Missing information can be filled in to enhance our internal representation of the object, for example although we might only be able to see three wheels, we assume that there is a fourth wheel.

In the hope of furthering our knowledge of this extremely difficult subject, the scope of the research was restricted to a small, yet useful, area.

Because of its remarkable processing power, some researchers began investigating using the brain as a model for research into Artificial Intelligence (AI). This research led to the field of neural networks.

Neural networks have the ability to classify patterns, and they have been used in artificial vision. One problem with using a neural network conventional pattern classifier is that it is not very good at classifying objects if they are not invariant. Therefore, it would not recognise a different view of the car. Even if the object was recognised as being a car, it could not determine the relationship between the various parts of the car.

If the vision system could analyse the image by considering the structure of the objects, the position or scale of an object would not present a problem. This approach is used in structural pattern recognition.

Taking ideas from neural networks techniques, neurobiology, and structural techniques, a neural network could be built solely for the classification of images in a translation and scale invariant manner.

1.1 RESEARCH OBJECTIVE

The object of this research is **to produce a neural network solution for translation and scale invariance**. As well as being able to classify objects, the network should be able to give a breakdown of the object. There should also be a facility for the network to recognise multiple objects in the same image.

A neural network approach was chosen because the above of its inherent parallelism. It was considered, that by retaining the locations of classified features in the image, a

network could be constructed that would form an internal representation of the image in a similar manner to the hierarchical structure of features in the image. By retaining the spatial relationship between the classified features, higher level features could be classified. By incorporating invariance into this feature classification, the network would have the property of size invariance. Interrogating the neurons could provide information about the structure of the image. Developing this type of neural network architecture was a very challenging proposition.

Neural networks can be used to classify objects in a translation and scale invariant manner. However, most neural network solutions to this problem use a conventional pattern classifying network that classifies the output from an invariance module.

Invariant vision neural network solutions have been proposed and implemented. However, their structure is significantly different to that of general neural network classifiers, in that the structure is only able to classify visual images. One such architecture is the Neocognitron. This has structural similarities with the network developed during this research. It is a hierarchical network that is tolerant to slight changes in the location of previous layer features. Although it is not a directly comparable, it has the closest architecture to the network described here and was used to provide a classification reference.

Neural networks can be taught in two general ways: either supervised on unsupervised. With supervised learning, the network is shown a number of input patterns that are expected to produce desired outputs. The network algorithms teach the network the new patterns. With unsupervised learning, the network is shown a number of patterns, and it has to learn these patterns without any knowledge from an external teacher. The human

visual system learns in an unsupervised manner and names are assigned to the learned objects.

Unsupervised learning is much harder to implement than supervised learning. This is because the network must determine for itself which neurons are to be taught. Developing robust algorithms to accomplish this would be problematic, but once established it would be easier to construct a training regime.

If supervised learning were to be used with an invariant network, the spatial location of the neurons must be considered. The teacher would have to indicate the location of the neuron to be taught or use some other method, for example ensuring that the salient feature was located at the centre of the image and only updating the neuron located at the centre of a layer.

The difficulty of producing the unsupervised learning algorithms is expected to be increased by the invariant nature of the network. This presents a challenging problem that would, if solved, lead to a network that is easier to train and to a better understanding of the network. The neural network developed during this research must be capable of learning in an unsupervised manner.

The research objectives are to be subject to the following constraints:

1. The vision system must be translation and scale invariant.

2. The vision system must be a neural network solution.

3. The vision system must learn in an unsupervised manner.

4. The vision system must be able to decompose the object into lower level features (sub-patterns).

The final system was expected to be complicated in layout, and involve considerable processing. It was expected that it would be slow in operation and there might be compromises with respect to image size and image shape. The final system contained a number of new features and should be seen as a basis for further research. It is not suitable for commercial applications.

The network is able to exploit the inherent parallelism of neural network techniques by being able to simultaneously classify multiple objects in an image. This is intended to make the network suitable as the basis for a 2D-scene analysis neural network [Bruce and Green 1990].

1.2 THESIS STRUCTURE

The structure of this thesis is as follows:

- Chapter 2 - Review of Natural and Artificial Vision

  This chapter contains a review of biological and artificial vision techniques. The problem of invariance is discussed. A discussion is also contained in this chapter, concerning the suitability of the various existing techniques in meeting the objectives is assessed, and the reasons why a new network was needed are justified.

- Chapter 3 - Shift Invariant Neural Network

  This chapter describes an early attempt to produce a specialised network for translation invariance. The details of this network have been superseded by later refinements, however the basic concepts are introduced.

- Chapter 4 - A Retinal Model and Grey Scale Classification

  This chapter describes the development of a simulation of a retina to test the translation invariant model with grey scale inputs. The results of testing are also given in this chapter.

- Chapter 5 - Scale Invariant Classification

  This chapter describes the structure of a neural network that is both translation and scale invariant, and is capable of structural decomposition. The results and discussion are also given in this chapter. A model of a Neocognitron was also built and used for comparison purposes.

- Chapter 6 – Discussion, Conclusions and Recommendations

  This chapter reviews the discussions in the previous chapters, and develops these into an overall discussion and conclusion.

- References

  The quoted references are listed.

- Appendix 1

  This appendix explains how the decision boundary between neurons can change with learning.

- Appendix 2

  This appendix explains how the choice of computer was made.

- Appendix 3

  This appendix contains the results for the shift invariant network.

- Appendix 4

  This appendix contains the results for the artificial retina and grey scale classification.

- Appendix 5

  This appendix contains the dendrite search and neuron calculation algorithms used in the scale invariant network.

- Appendix 6

  This appendix contains the results for the scale invariant network and the Neocognitron.

2.0 REVIEW OF NATURAL AND ARTIFICIAL VISION

2.1 CHAPTER CONTENTS

The aim of this research was to investigate a neural network solution for position and scale invariant vision, and inspiration from both artificial and biological techniques was sought. This requires an understanding of both techniques.

The biological approach was investigated because it is a superb system, and has outstanding performance in a number of areas including invariance to translation, brightness and scale. Although inspiration from biological techniques was sought, it does not mean that the research objective was to model biological vision and no claim is made to the biological plausibility of the networks developed during this research. Instances where biological techniques have been used are highlighted in this chapter.

In the review of existing computer vision and neural network techniques, the suitability of these techniques in meeting the research objectives was assessed.

2.2 OVERVIEW OF MAMMALIAN VISION

The mammalian vision system, being the most advanced, is now briefly described. This system, like all neural processes, is built up from large number of neurons. The behaviour of these neurons is only slightly understood but some aspects of their behaviour have been established.

2.2.1 DESCRIPTION OF A NEURON

A neuron is a cell which is specialised in transmitting information. The following discourse on the structure of a neuron is more fully discussed in books by Richard Thompson [Thompson 1985], M. B. V. Roberts [Roberts 1974], and [Fischbach 1992]. Neurons have a characteristic appearance with a cell body, from which a number of fibres emanate. One of these fibres is called the axon, the output of the neuron, and is connected to other neurons, muscle cells, or gland cells. As only information processing is being considered, only neuron - neuron connections will be considered. The synaptic terminals connect with dendrites of the other neurons at the dendrites or the cell body, forming synapses. There are two types of synapses, chemical and electrical. Electrical synapses are considered to have evolved before chemical synapses. They cannot readily change their processing characteristics, and are therefore not amenable to learning. As the axon approaches the target cells it branches into a number of smaller axons, each ending in a specialised terminal called the synaptic terminal.

The other fibres are known as dendrites, and can be regarded as the neuron's input. Axons from other neurons connect their synaptic terminals with the dendrites and the cell body at junctions known as synapses. Neurons can have thousands of synaptic connections. A simplified drawing of a neuron is shown in Figure 2.1.

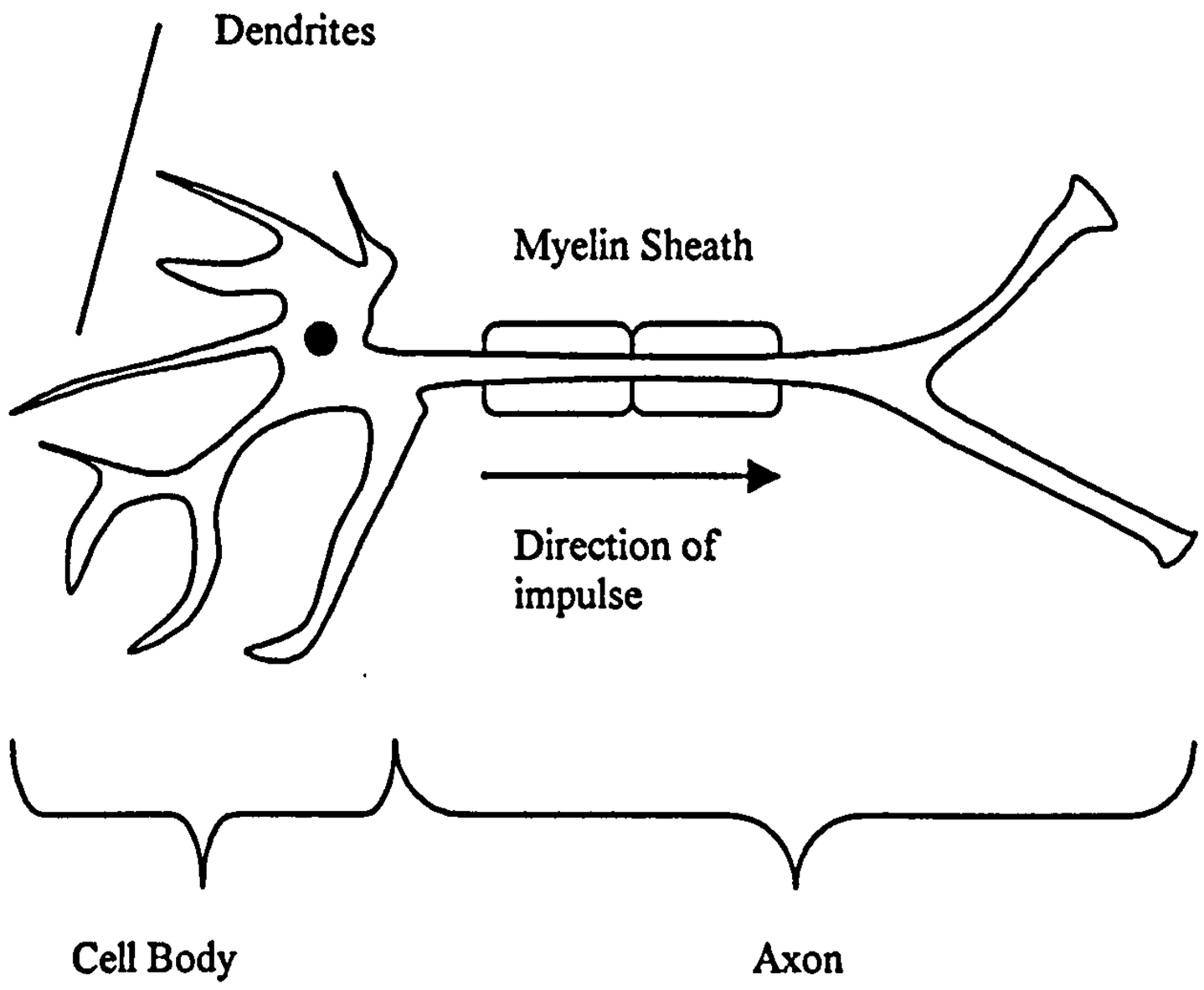


Figure 2.1 Simplified neuron

The electrical activity of neurons is produced by cell ions; K^+, Na^+ and Cl^- and internal protein molecules (P^{2-}). These ions are distributed across the cell membrane that utilises non-gated and gated ion channels and active transport mechanisms. Under the influence of both diffusion and electrical attraction an equilibrium voltage of about –70 mV is maintained across the membrane of the axon. This equilibrium voltage can be predicted by the Goldman equation [Thompson 1985].

An increase in voltage in the axon can open the Na^+ and K^+ ion channels that will cause a sudden change in cell permeability leading to a sudden change in potential to about 50 mV. Ion channels start to close and the potential starts to return to the equilibrium potential. However the rise in voltage will cause adjacent ion channels to open and the spike will traverse along the axon.

Chemical synapses have a gap of about 20 nm between neuron membranes, which is known as the synaptic cleft. Information is transferred across the synaptic cleft in one direction only. The neuron that sends the signal is known as the pre-synaptic neuron, and the receiving neuron is known as the post-synaptic neuron. This signal is a chemical messenger called a neurotransmitter. These neurotransmitters are stored in containers called vesicles located in the synaptic terminals. There are three differently shaped vesicles, with their function classified by the effect that the neurotransmitter has on the post-synaptic neuron. The neurotransmitters in spheroid vesicles are thought to increase the activity of the post-synaptic neuron (excitatory), flat vesicles decrease the activity of the post-synaptic neuron (inhibitory), and the function of the neurotransmitters in the dense-core vesicles has not yet been discovered.

A synapse can form on three places on the neuron: on a dendrite spine, on a dendrite and on the cell body. The synapses that form on the dendrite spines and dendrites are considered to be excitatory whereas the synapses on the cell body are considered to be inhibitory.

Many neurotransmitters (about fifty) have been discovered and these can be grouped into two large superfamilies. The first family of neurotransmitters affects the membrane potential, and the second family indirectly modifies the behaviour of the ion channels using second-messengers (described later). The membrane potential can be affected in an excitatory or inhibitory manner. Release of the neurotransmitters is initiated by the arrival of the action potential at the synaptic terminal. Voltage gated calcium ion (Ca^{2+}) channels are opened briefly, allowing Ca^{2+} ions into the synaptic terminal. This triggers the vesicles to fuse with the pre-synaptic membrane and release the neurotransmitters; a process called exocytosis.

In the case of an excitatory synapse, when the neurotransmitters reach the post-synaptic membrane they briefly trigger the opening of closed chemical gated Na^+ channels by attaching themselves to receptor molecules in the membrane. This causes the membrane to depolarise. The strength of this depolarisation decreases with distance from the synapse.

The axon depolarisation caused by the firing of one synapse is not sufficient to trigger an action potential. However, if a number of synapses fire, the combined depolarisation can be enough to trigger an action potential. This summation of the effect of several synapses is called spatial summation. Similarly if one synapse is repeatedly fired, it can also trigger an action potential; this is caused temporal summation. A neuron is continuously summing information over time and space, this sum being known as the excitatory post-synaptic potential (EPSP). When the EPSP at the axon initial segment reaches the electrically gated ion channel threshold voltage, an action potential is generated.

There is a similar mechanism initiated by the firing of inhibitory synapses. The effect of firing inhibitory synapses is spatially and temporally summed, giving an inhibitory post-synaptic potential (IPSP) at the axon initial segment. The EPSP and IPSP potentials do not sum in a simple linear manner; each neuron is a sophisticated computer [Fischbach 1992].

Simplified models of neurons are used in the field of artificial intelligence and are discussed later in this chapter.

It has been found that inhibition has a stronger effect than excitation in the generation of an action potential. This is because the inhibiting Cl^- ions act as a shunt to prevent the positive charge of the exciting Na^+ ions reaching the axon initial segment. In addition, the inhibitory synapses tend to be closer to the axon initial segment than the excitatory

synapses, so they have a greater effect during the spatial summation. This mechanism is used in the size invariant network described in Chapter 5.

Habituation is the behavioural response to a repeated stimulus [Thompson 1985]. This can easily be observed in a sea anemone; if it touched it is will contract, but repeated touching has less of an effect. This is thought to be caused by synaptic depression, where less neurotransmitter is released on repeated stimuli. This mechanism is used in the unsupervised learning described in Chapter 3. In contrast, sensitisation is the behavioural response where a stimulus causes an increase in response, for example a sudden loud bang could make somebody be more alert.

Learning is the process by which we acquire new knowledge [Kandel and Hawkins 1992]. It has been observed that the connections between neurons can become altered during learning [Alkon 1989], where there appears to be focusing of the synapses. Some branches may atrophy or reduce in size, and others may increase in size. The magnitude of the branching volume is related to the effectiveness or strength of the synapse. A mechanism for atrophying synapses is used in the neuronal model described in Chapter 3.

The biological basis for this process was first proposed by Donald O. Hebb in 1949 [Hebb 1949]. In his book he proposed that the strength of a synapse is increased following the simultaneous firing of the synapse and the post-synaptic neuron, this has been subsequently referred to as Hebb's Law. Although this principle was not mathematically stated, equations have been based on this principle and have been used in learning algorithms, giving Hebbian Learning.

In the case of Pavlovian Learning [Roberts 1974] which is a form of associative learning, a local-interaction model has been proposed [Alkon 1989], where the firing of the post-synaptic neuron is not required for learning. The interaction of the adjacent unconditioned stimulus synapse and conditioned stimulus synapse increases the synapse's strength.

2.2.2 MAMMALIAN VISUAL SYSTEM

The mammalian visual system will now be considered, starting from the retina and ending with the organisation of the visual system in the brain. Images are formed on a photosensitive region at the back of the eye called the retina. Electron microscope images of the retina have helped to deduce its structure, and a typical view through the mammalian retina is shown in Figure 2.2.

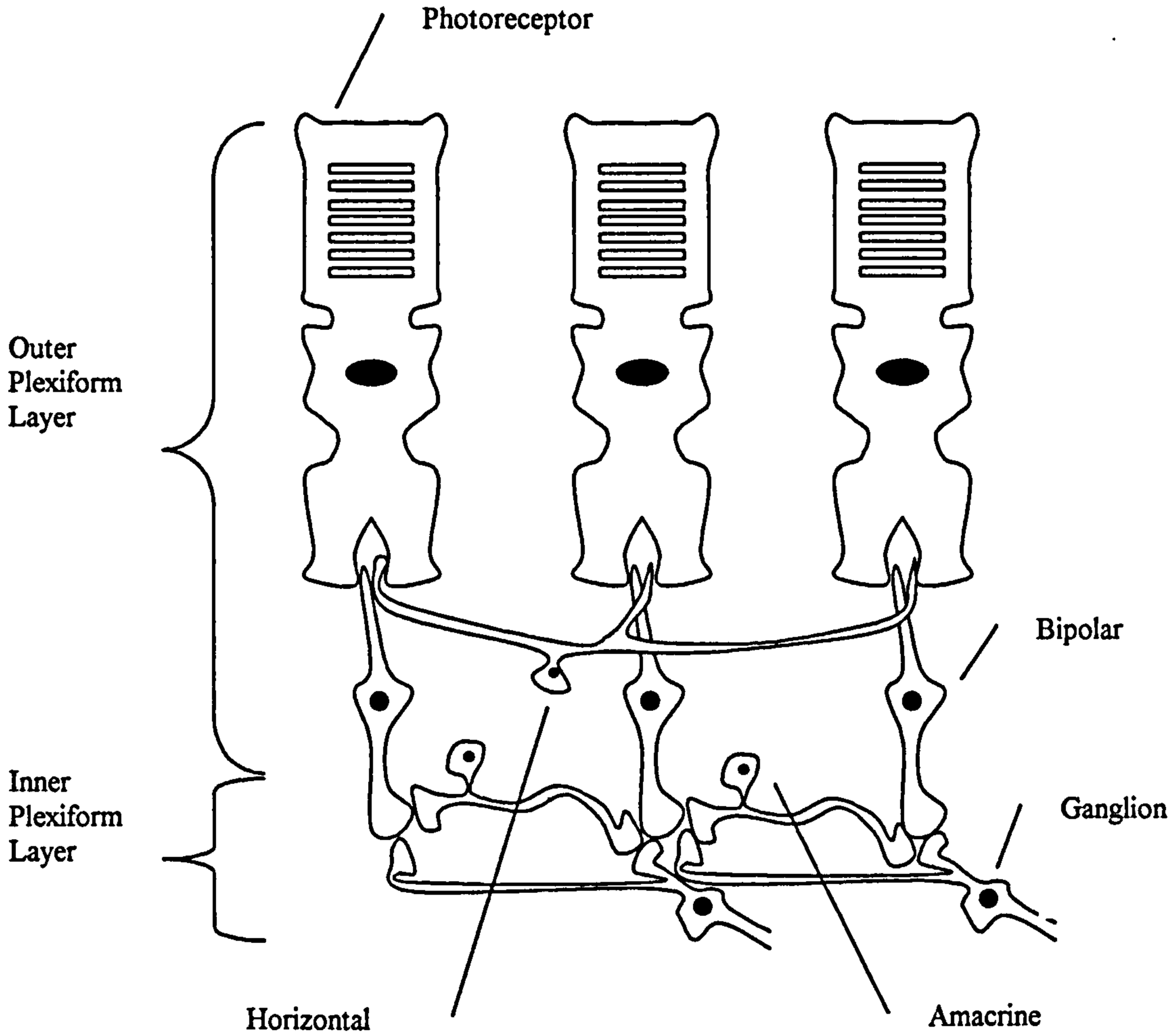


Figure 2.2 Simplified section through a retina

The light sensitive cells are called the photoreceptor cells, and they give a membrane hyper-polarisation that is roughly proportional to the log of the light intensity. Beyond the photoreceptors are two layers of neurons, each layer consists of neurons with pathways at right angles to each other.

Connecting with the photoreceptors are three horizontal and bipolar neurons; these neurons form the outer plexiform layer. These neurons connect with the amacrine and ganglion neurons that form the inner plexiform layer. The ganglion neurons send the information to the brain. Research on their behaviour has identified a number of classes of ganglion neurons, known as X, Y, and W. The retina structure is revisited in Chapter 4.

The arrangement of the right angled neuronal pathways in the retina makes a ganglion neuron sensitive to the light falling on a number of photoreceptors; this region of sensitivity is called the receptive field. One type of receptive field is the concentric field, in which there is a circular region of photoreceptors which either excite or inhibit the ganglion when a point of light only falls on this region. Surrounding this circular region is an annulus of photoreceptors that have the opposite effect on the ganglion neuron.

A ganglion with a centre-on response is excited when a dot of light only falls on the centre, and is inhibited when a dot of light only falls on the surround. Conversely, a centre-off ganglion neuron is inhibited when a dot of light only falls on the centre region, and is excited by a dot only falling on the surround.

One form for the concentric receptive fields has been proposed [Marr 1982] in which the output from a ganglion neuron can be modelled by convoluting the receptor output by a 2-

dimensional Gaussian function, and then performing a Laplacian operation. This can be approximated by taking the difference of two Gaussians (DOG) of the receptor output.

The X ganglion neurons exhibit a centre-off or centre-on response. These neurons become excited by a boundary positioned across the receptive field. The response to a moving sinusoidal grating is linear, and gives a sustained response to a stationary sinusoidal grating.

Y ganglion neurons have a non-linear response to sinusoidal gratings, giving a constant spike rate for a moving grating. The response to a stationary grating is transient.

The W ganglion neurons have slowly conducting axons, and can be classified into a number of different types; there are two common ones and a number of rarer types. One common type is called on-off or local edge detectors, these respond to a spot of light moving in and out the centre of the receptive field. The other common type of W ganglion responds to a moving spot of light; many of these neurons have directional sensitivity, giving their maximum response to a preferred direction of movement.

Mechanisms have been proposed to account for this directional sensitivity. The biological mechanism proposed in [Poggio and Christof 1987] uses local shunting inhibition, where the effect of the inhibitory neurotransmitters does not hyper-polarise the neuron, but instead maintains the membrane potential close to the resting potential.

Movement of the object is defined by the firing of two different photoreceptors, with one photoreceptor firing before the other to denote the preferred or null directions. One photoreceptor initiates an excitatory potential and the other initiates an inhibitory

potential. Considering the length and widths of the signal paths from the photoreceptors to a ganglion neuron, and by modelling the movement of the neuron potential as the current flow in a transmission line, the arrival time of the potentials at the ganglion cells could be determined.

The synapses on the ganglion dendrite are very close together with the inhibitory synapse closer to the cell body. When the order of firing of the photoreceptors is such that it mimics movement in the preferred direction, the excitatory potential reaches the ganglion neuron before the inhibitory potential and the ganglion fires. However when simulating the null direction, the action potentials reach the ganglion at the same time and the inhibitory signal vetoes the excitatory signal. This illustrates how signal timing and position of synapses can solve problems. A modification of this system was used in the size invariant network described in Chapter 5.

A motion-tracking model has been developed from retinal techniques in [Eeckman et al. 1989].

The relative proportion of W ganglion neurons varies considerably with species. Frogs have a much higher proportion than cats, which in turn, have more than primates. The frog retina has more processing capability that the retina of primates. It has been discovered that the frog's retina contains one class of ganglion neuron that acts as a bug detector, and only responds to an irregularly moving small dark object. In higher animals, more of the visual processing is carried out in the brain, facilitating the improvement of the visual system by learning.

The size of the receptive fields is at a minimum at the fovea centralis of the retina and increase in size towards the periphery. Therefore, the resolution of the retina is highest at the fovea.

It has been argued [Wilson 1985], that this logarithmic retino-cortical mapping can perform a scale invariant transform. Provided that the image can be rotated and scaled about a centre, the rotation and scaling effects can be transformed into translation. This transformation is not invariant to translation.

Returning to the afferent pathway of the visual system, a classical model has been developed from pioneering work by Hubel and Wiesel [Hubel and Wiesel 1968]. This model suggested a hierarchical structure starting with simple neurons (cells). Simple cells have receptive fields in which the boundaries between excitatory and inhibitory regions are straight lines. Examples of simple cell receptive fields are shown in Figure 2.3.

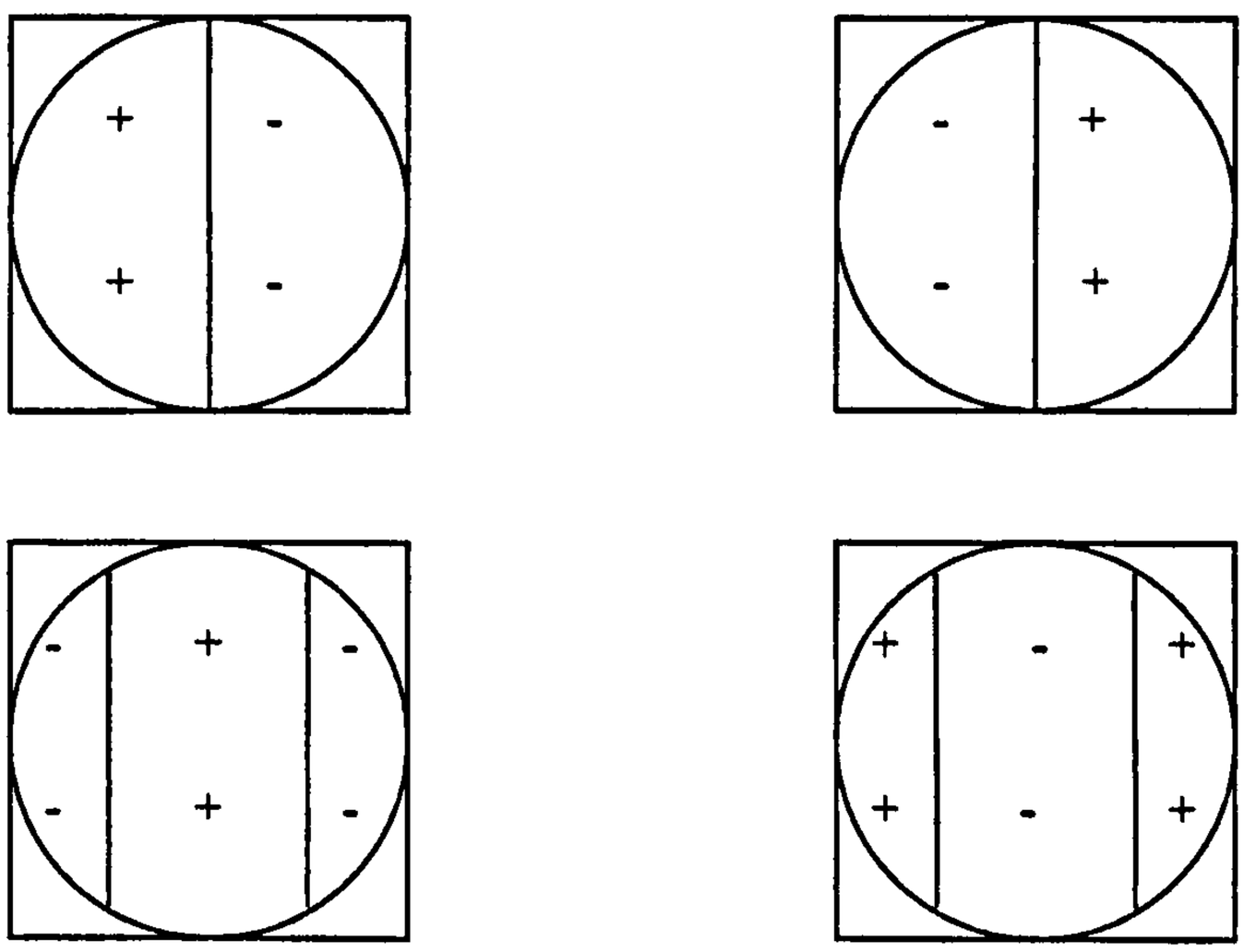


Figure 2.3 Examples of simple cell receptive fields

Simple cells respond to edges and express orientation preference. Cells with the same orientation preference are arranged in columns. Presenting edges more than about 20° away from the preferred orientation will significantly reduce the cell's output. Cells that respond to more complicated features were used as primitive detectors in the size invariant network described in Chapter 5.

The output from the simple cells is fed into complex cells, which become active when a simple cell in its receptive field becomes active. A third type of cell called hypercomplex has been proposed, which is similar to a complex cell but elicits its greatest response to an edge not extending beyond the receptive field. Later studies suggested that the behaviour of complex cells is not as clear cut as was initially thought [Spitzer and Hochstein 1988], and it has been proposed that complex cells non-linearly sum their inputs. Models of simple and complex cells are used in the Neocognitron [Fukushima et al. 1983] to classify images.

Studies of the visual cortex have revealed that the visual system is highly organised, and consists of a number of visual areas. Although the number and boundaries of these areas have not been accurately determined, progress has been made. Figure 2.4 shows a proposed model for the macaque monkey visual cortex [Maunsell and Newsome 1987], where region V1 is the area of the cortex described above. The cells in region V2 have larger receptive fields than the cells in V1, and have an orientation preference. Other regions have other dominant properties, for example the cells in region MT are strongly selective for speed and direction of motion, whereas V4 is sensitive to colour. The higher regions have larger receptive fields, some of which extend across the visual field.

The organisation of the visual cortex combined with the massive parallelism has produced a superb visual processing system. This can detect multiple objects and is invariant to object position, brightness and size. In addition to the classification performance, it is able to learn by unsupervised learning under imperfect conditions. The highly ordered visual cortex suggests significant pre-wiring during foetal development under the influence of the genes [Thompson 1985].

The highly specialised structure of vision systems is revisited when describing the size invariant layer in Chapter 5.

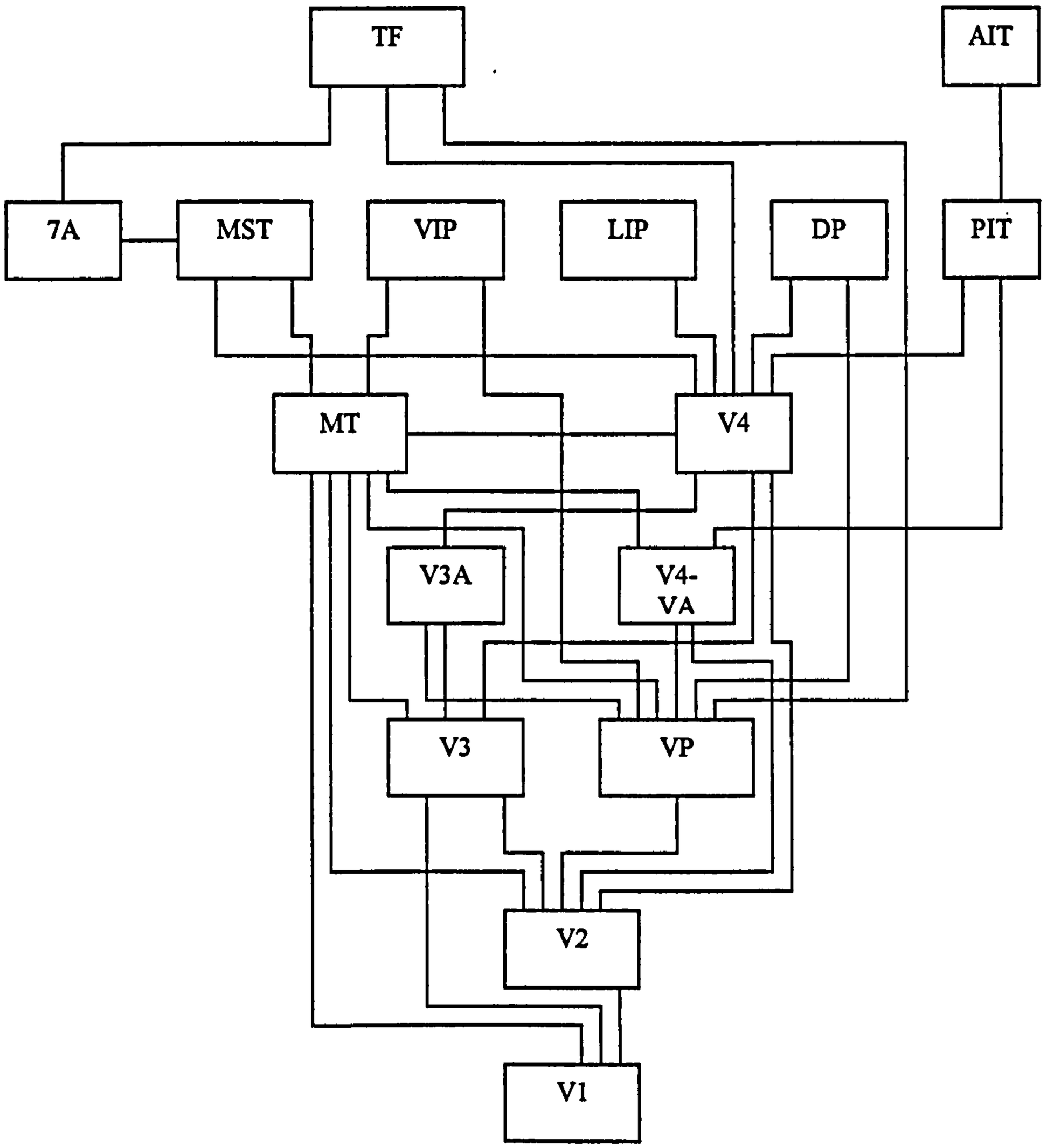


Figure 2.4 Proposed model for the macaque monkey visual cortex

2.3 REVIEW OF ARTIFICIAL VISION

Artificial vision uses sophisticated electronics and/or algorithms to accomplish visual perception. Normally this visual perception falls short of the performance of biological systems, so the scope is limited to solving specific visual problems. Other reviews of invariance computer vision have been made [Wechsler 1987], [Wood 1996] and [Wood and Shaw-Taylor 1996]. Many techniques have been devised, but the techniques most relevant to this research will be discussed here.

Most visual systems capture the visual information using a video camera; signals from the camera are digitised to form discrete picture elements or pixels. The digitised values are stored as complete image(s) in a frame buffer, or frame grabber, and then transferred to computer memory.

The image processing algorithms operate on the image stored in computer memory. However some vision systems can have extra inputs; an example is character recognition software that senses the motion of the pen. This extra information can improve the recognition process. Sensing the real-time characteristics of the image is called *on-line* processing (this is known as On-Line Character Recognition or OLCR), whereas processing the image after it has been formed is termed *off-line* processing (also referred to as Optical Character Recognition or OCR) [Hutton 1989].

OLCR systems record a time sequence of x and y values, although filtering may be required to remove the effects of noise, processing is usually simpler and faster than for OLCR. Using time as an input parameter allows the sequence of pen movements to be determined quickly, and the duration of various pen strokes to be estimated. In practice, it has been found that OLCR has a higher recognition rate than OCR. OLCR can be used for

cursive script characters [Hoffman et al. 1993]. OCR techniques are reviewed in [Govindan and Shivaprasad 1990].

2.3.1 TEMPLATE MATCHING AND ENHANCEMENTS

Template matching is a simple technique in which templates are constructed from sub-images containing salient features. A number of templates can be stored in the system, describing different views of the same object and different objects. Classification is achieved by testing each template at every possible location on the image, and the template with the best fit is used to denote the object. This can be easily described mathematically using vector notation.

A vector $\mathbf{w}$ describes a point on the image, and $\mathbf{x}$ is a vector that describes a point on the template. The pixel value $f(\mathbf{w})$ describes the intensity for any point $\mathbf{w}$ on the image, and the pixel value $t(\mathbf{x})$ describes the intensity for any point on the template.

Subtracting one pixel value from another can give either a positive or a negative difference. One method of ensuring that the differences in pixel values will be positive is to square the difference. Summing the squared differences for the template gives the following error:

$$d^2(\mathbf{y}) = \sum_{\mathbf{x}} \left(f(\mathbf{x}+\mathbf{y}) - t(\mathbf{x})\right)^2 \qquad 2.1$$

$$d^2(\mathbf{y}) = \sum_{\mathbf{x}} \left(f^2(\mathbf{x}+\mathbf{y}) - 2f(\mathbf{x}+\mathbf{y})t(\mathbf{x}) + t^2(\mathbf{x})\right) \qquad 2.2$$

Considering the co-ordinates as a vector quantity, the quantity $d(\mathbf{y})$ is the L2 Euclidian norm. Determining the minimum $d^2(\mathbf{y})$ from a number of templates will give the location and type of feature present in the image. As all terms in the summation are ≥ 0, the minimum possible value is 0. Considering a simplified analysis, as the value for $t^2(\mathbf{x})$ is constant, and the value for $f^2(\mathbf{x}+\mathbf{y})$ can be regarded as varying slowly (and treated as a constant), so the $d^2(\mathbf{y})$ term is minimised by maximising

$$\sum_{\mathbf{x}}(f(\mathbf{x}+\mathbf{y})t(\mathbf{x})) \qquad 2.3$$

This term is the cross-correlation between $f(\ )$ and $t(\ )$, and represents the convolution of $f(\ )$ and $t(\ )$.

This term must be calculated for all positions of $\mathbf{y}$, and for all templates to determine the template with the largest value. These calculations can be time-consuming but can be speeded up by transforming the image and the template. Considering Fourier theory, the Fourier Transform of the convolution of two functions can be considered to be the product of the Fourier Transform of each function.

The cross-correlation can be determined by taking the Fourier Transform of $f(\ )$ and $t(\ )$. Multiplying the two transforms gives the Fourier Transform of the convolution of $f(\ )$ and $t(\ )$, then calculating the Inverse Fourier Transform of this product gives the cross-correlation.

Calculating the Fourier Transform can be notoriously slow, but using the Fast Discrete Fourier Transform, this method can be faster than the spatial cross-correlation method, particularly for larger templates [Boyle and Thomas 1988].

Template matching techniques can be enhanced to consider the effects of the noise components, leading to the field of matched filters [Dessimoz at al. 1984].

Template matching is invariant to translation but not scale. It normally tries to match the whole image and therefore does not have the capability to determine the structure of the image.

2.3.2 AFFINE INVARIANT SHAPE MATCHING

Planar objects seen by two cameras at different positions can be related by affine transformations. If the reference image was taken by one camera, and the other camera is used for recognition purposes, dominant point matching can be used to classify the objects. A set of three boundary points should be sufficient to match the two objects.

Research has been involved in trying to provide a unique and invariant point set for each specific object. Difficulties have arisen due to the problems of capturing the image, for example including noise, and using certain classes of object that lack dominant points.

The solution can be reformulated as a search process, involving matching three non-collinear points on the reference contour with three reference points on the scene contour. After calculating the transformation parameters between the two sets of co-ordinates, the two contours can be aligned on top of each other. The correlation between the two overlaid contours can be determined. Calculating the correlation between all reference contours and

the scene contour for all possible sets of points on the scene contour, will result in the best match being determined. This search would be prohibitively expensive in terms of computing time, and AI techniques exist to speed up this search process. If we are looking for the smallest solution, for example the amount of mismatch between the two contours, when the current value is surrounded by larger values, a minimum value has been found. It is possible for there to be a number of minimum values in the search space topology, of which only one is the best value. This is called the global minimum and the others are called local minima. The search algorithm must strive for the global minimum solution.

Genetic Algorithms (GA) have been used [Tsang 1995] to search for the best match between the reference and scene contours. Genetic Algorithms and a later innovation called Genetic Programming (GP) have been used for solving many not-polynomial (NP) problems for example planning maintenance of railway track [Grimes 1995].

The affine transformation between the two contours can be related by:

$$\begin{bmatrix} x_{u;i} \\ y_{u;i} \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x_{r;i} \\ y_{r;i} \end{bmatrix} + \begin{bmatrix} e \\ f \end{bmatrix}_{i=0,1,2..n} \qquad 2.4$$

Where,

$x_{u;i}$, $y_{u;i}$ are the co-ordinates of a point on the scene contour $x_{r;i}$, $y_{r;i}$ are the co-ordinates of a point on the reference contour

a, d are the scaling terms

b, c are the rotation terms

e, f are the translation terms

Three points on each reference and scene contour are used for matching purposes. Putting each set of matching points in Equation 2.4 gives three equations which can be used to give the above transformation terms. Knowing these terms, the reference contour can be transformed to try to match the scene contour.

The match between the transformed reference and scene contours is given by:

$$MS = \left(1 - \lambda_1 \frac{A\left(O_{scene} / O'_{ref}\right)}{A\left(O_{scene}\right)}\right)\left(1 - \lambda_2 \frac{A\left(O_{ref} / O'_{scene}\right)}{A\left(O_{scene}\right)}\right) \qquad 2.5$$

Where

$A(\mathrm{O})$ represents the area of image O

$A(\mathrm{X} / \mathrm{Y})$ represents the area of the region contained in X but not Y

The first term of the product reflects the amount that O_{scene} has in common with O_{ref}, and will return 1 for a perfect match. However a value of 1 would also be returned if O_{scene} was contained within O_{ref}. To prevent this, the second term is added.

The coefficients reflect the strength of each term, with λ_1 having a higher value than λ_2.

With no prior knowledge of the relationship between the reference and scene object, the system was able to classify unknown images in about 1 minute using a 50Mz 80486 based PC [Tsang 1995].

Affine shape matching can be considered to be an enhanced template matching system that can match reference images and input images that are subject to affine transforms, so it is also scale invariant. Similarly to template matching, it is not able to determine the structure of the image.

2.3.3 HOUGH TRANSFORM

The Hough Transform is a method of identifying geometrical features in an image. There is an accumulator array for each feature that is being sought, that has one dimension for each unknown parameter.

The array is initialised by setting all values to zero. The image is searched for the feature. Where a possible feature has been found, the array element associated with the parameter values is incremented. After searching the image the array is searched for local maxima, which represent evidence for the features.

The Hough Transform is useful for detecting simple shapes or features, particularly if there is prior knowledge of these features, for example the radius of a circle. It also has the property of being able to identify a feature if only part of it is shown in the image. This method is naturally invariant to translation. Unfortunately, the method can give ambiguous results.

Further processing is required to determine the structure of the image.

2.3.4 FOURIER TRANSFORM

Fourier developed the mathematical principle in which a periodic function can be represented by a sum of sinusoidal waveforms and a constant. By increasing the time period T, the harmonic spacing reduces until the Fourier coefficients merge into a continuous spectrum. This is known as the Fourier Transform (FT) [Newland 1981] and is given by:

$$X(\omega)=\frac{1}{2\pi}\int_{-\infty}^{\infty}x(t)e^{-j\omega t}dt \qquad 2.6$$

The position of the 2π in Equations 2.6 can vary between authors, this coefficient is not important in the subsequent further analysis and will be dropped.

The above equation is for a one dimensional FT, Equation 2.6 can be extended for two dimensions giving the following equation [Wechsler 1987]:

$$F(u,v)=\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}f(x,y)e^{-j2\pi(ux+vy)}dxdy \qquad 2.7$$

A two dimensional form given by Equation 2.7 can be used to determine the frequency components of continuous signals in two directions. When considering an image, a two dimensional pattern, the image is quantised and a discrete form of 2.7 is used:

$$F(k,l)=\frac{1}{MN}\sum_{n=0}^{N-1}\sum_{m=0}^{M-1}f(n,m)e^{-j2\pi\left(\frac{kn}{N}+\frac{lm}{M}\right)} \qquad 2.8$$

This form is known as the discrete Fourier Transform or DFT.

Applying a translation to the image by the amount $\mathbf{d} = (n_0, m_0)$ gives

$$F_t(k,l) = F(k,l)e^{-j2\pi\left(\frac{kn_0}{N}+\frac{lm_0}{M}\right)} \qquad 2.9$$

Taking the magnitudes of both sides of 2.9 gives

$$|F_t(k,l)| = |F(k,l)| \qquad 2.10$$

This gives the property that the magnitude of the frequency components is invariant to translation, and the phase contains the translation information.

The above method for calculating the DFT is numerically very intensive. The fast Fourier Transform (FFT) has been devised to reduce the number of total calculations required to produce the FT. It works by partitioning the overall calculation into calculations at half resolution, quarter resolution, eighth resolution, etc, and combining these partitioned calculations into the total FT [Newland 1981]. The reduced number of calculations gives a faster and more accurate result.

This method considers the whole image and the structure of the image is not determined.

2.3.5 MELLIN TRANSFORM

Although the FT is invariant of translation, it is sensitive to changes in scale. Viewing an object at a non-zero aspect angle giving profile compression can produce this scale change. Expanding the compressed profile can introduce noise. The Mellin Transform (MT) described here can help in image classification because of its scale invariance property.

Given a function $g(t)$, $t \geq 0$ the continuous one-dimensional MT can be defined as [Zwicke and Kiss 1983]:

$$G(s) = \int_0^\infty g(t) t^{s-1} dt \qquad 2.11$$

Using exponential distortion, $t = T^x$, gives

$$G(s) = T^s \int_{-\infty}^{\infty} g\left(Te^x\right) e^{xs} dx \qquad 2.12$$

The scaling term becomes transformed to a translation term, which is further transformed to a phase term by the FT calculation.

Because the lower limit of the integration is -∞, a correction term must be added to the value for $G(s)$. This term is known as $g(0)$ correction.

For the discrete implementation, if the FFT method is used to calculate the MT, this transform is known as the fast Mellin Transform or FMT.

The MT is not insensitive to translation, and the synchronisation of two different images would have to be carried out before the transform was calculated. The property that the FT is invariant to translation can be used to solve this problem. By producing the FT of an image, then producing the MT of the magnitude of the FT results in a process that is invariant of both translation and scale. The resultant process is known as the Fourier-Mellin (FM) transform. The exponential sampling of the FT output acts as a low pass filter, and the higher order FM terms have low magnitude making classification difficult.

It is possible to calculate the MT directly without using the FT, this is known as the direct Mellin transform or DMT. This formulation for the DMT is exact for sampled data, and $g(0)$ compensation is not required. As the DMT relies on the difference between the values of adjacent data points, this formulation has implicit differentiation.

The FM transform has low pass filtering characteristics that can make classification using high order frequencies difficult. Classification can be improved using a modified form of the DMT, called the modified DMT or MDMT. This involves multiplying the DMT by the frequency term as

$$G_M(\omega) \triangleq -j\omega G(\omega) \qquad 2.13$$

The MDMT has been used to give good discrimination between frigate and cruiser profiles [Zwicke 1983].

[Zwicke 1983] has summarised the following characteristics of the MT, as shown in Table 3.1.

Similarly to the Fourier Transform, the structure of the image is not determined.

Acronym	Transform	Operations	Comments
FMT	Fast Mellin	1) Exponential sampling 2) Interpolation 3) FFT 4) g(0) correction 5) Magnitude	1) Most common implementation 2) Scale invariant 3) Requires detection of profile initiation 4) Computational burden is N ln N
DMT	Direct Mellin	1) Pre-compute and store coefficient matrix 2) On-line data subtraction 3) Matrix-vector multiply 4) Multiply by 1/ω	1) No g(0) correction or interpolation required 2) Scale invariant 3) Requires detection of profile initiation 4) Computational burden in N
F-M	Fourier-Mellin	1) FFT 2) Normalise 3) Magnitude 4) FMT or DMT	1) Scale invariant 2) Translation invariant 3) Low-pass filtering characteristic 4) Increased spectral detail through zero filtering
MDMT	Modified Direct Mellin	1) DMT exclusive of Step 4 OR 2) FMT multiplied by ω	1) Scale invariant 2) $\omega G(\omega) \leftrightarrow t\frac{dg(t)}{dt}$
F-MDMT	Fourier-Modified Direct Mellin	1) F-M with MDMT used in Step 4	1) Scale invariant 2) Translation invariant 3) Compensates for low-pass characteristic

Table 3.1 Transform characteristics adapted from [Zwicke 1983]

Invariance transforms can also be represented by group theory [Lenz 1990], and can be used for translation, rotation and scale invariance using a Lie transformation group [Squire 1996].

2.3.6 MOMENTS

Invariant pattern recognition can be carried out using moments of area, which have been determined for the pattern found in the image. For a two dimensional continuous shape function, $f(x,y)$, the moment of order $p+q$ is defined by;

$$I_{pq} = \iint x^p y^q f(x,y) dxdy \qquad 2.14$$

Where $p, q = 0, 1, 2, 3, ...$

From the above relationship I_{00} is the area of the sub-image, I_{10} is the first moment of area (centroid or $\bar{x}$) along the x-axis, and I_{01} is the first moment of area (centroid or $\bar{y}$) along the y-axis.

The reference point or origin about which the moments are taken is important. It is convenient to take moments using the centroid as the origin; these moments are called central moments.

For a continuous function, central moments are defined by

$$I_{pq} = \iint (x-\bar{x})^p (y-\bar{y})^q f(x,y) dxdy \qquad 2.15$$

Similarly for the discrete case they are defined by

$$I_{pq} = \sum\sum (x-\bar{x})^p (y-\bar{y})^q f(x,y) \qquad 2.16$$

The use of central moments means that the translation components have been removed. To remove the effect of scale, a set of seven normalised central moments have been defined [Hu 1962].

Instead of considering the moments in terms of Cartesian co-ordinates, the central moments can be calculated in terms of polar co-ordinates. This allows greater flexibility in the choice of weighting function.

Moments can also be used to classify rotated images. The 2nd order central moments for a rotated pattern, can be expressed in terms of u_{02}, u_{11}, and u_{20}. It is possible to determine the angle of rotation for which the principle moments reach maximum values.

Two angular solutions, 90° apart, are given denoting the principal axes for the image. The magnitude of these principal second moments of area can be used for classification purposes, and the angles of the principal axes gives the degree of rotation.

This method does not consider the structure of the image.

2.3.7 STRUCTURAL TECHNIQUES

Image classification is normally based on the concept of having a number of known prototypes, and comparing an unknown pattern against these prototypes. The prototype that is the most similar to the unknown pattern denotes the classified pattern. Classification can be described as using decision/theoretic and syntactic/structural approaches [Thomason 1990]. Examples of the former technique are given by the techniques described earlier in Sections 2.3.1 to 2.3.6. Syntactic and structural pattern

recognition considers how the sub-patterns (or elements or components) of the image are related to each other.

By defining a pattern in term of the relationship between sub-patterns, the distance between these sub-patterns is not considered, allowing classification independent of translation and scale.

The methods used to classify patterns using the structure broadly fall into two categories: syntactic and structural, where syntactic analysis is a procedure for determining structural similarity using a grammatical approach [Bunke 1990a]. Structural analysis uses non-grammatical approaches, and a hybrid of the two approaches can be used.

Syntactic methods use a grammar to represent the structure of the pattern. String grammars are defined by [Thomason 1990, Bunke 1990a]. An alphabet, A, is a finite set of symbols. A word x over A is a sequence of symbols:

$$x = a_1 \cdots a_n \qquad 2.17$$

Where

$$a_i \in A;\ i = 1, \cdots, n$$

The set of all words, including the empty word ε, over an alphabet A is denoted by A^*.

The set of all words excluding the empty set is given by A^+, where

$$A^+ = A^* - \{\varepsilon\} \qquad 2.18$$

A formal grammar is defined by

$$G = (N, T, P, S) \tag{2.19}$$

Where,

N is a finite set of non-terminal symbols.

T is a set of terminal symbols.

P is a finite set of production, or rewriting rules.

S is the initial starting symbol, $S \in N$.

There can be no common items in N and T, so

$$N \cap T = 0 \tag{2.20}$$

The vocabulary is the union of set N and T, so

$$V = N \cup T \tag{2.21}$$

Each production rule is of the form

$$\alpha \rightarrow \beta \tag{2.22}$$

Where α is the left-hand and β is the right-hand, the production rule means that α may be rewritten as or replaced by β.

A language generated by grammar G is given by;

$$L(G)=\left\{x \middle| x \in T^*, S \xrightarrow{*} x\right\} \qquad 2.23$$

Syntactic recognition tries to determine whether the word is contained in the alphabet, or

$$x \in L(G) \qquad 2.24$$

A classifier using syntactic pattern recognition has the block diagram shown in Figure 2.5.

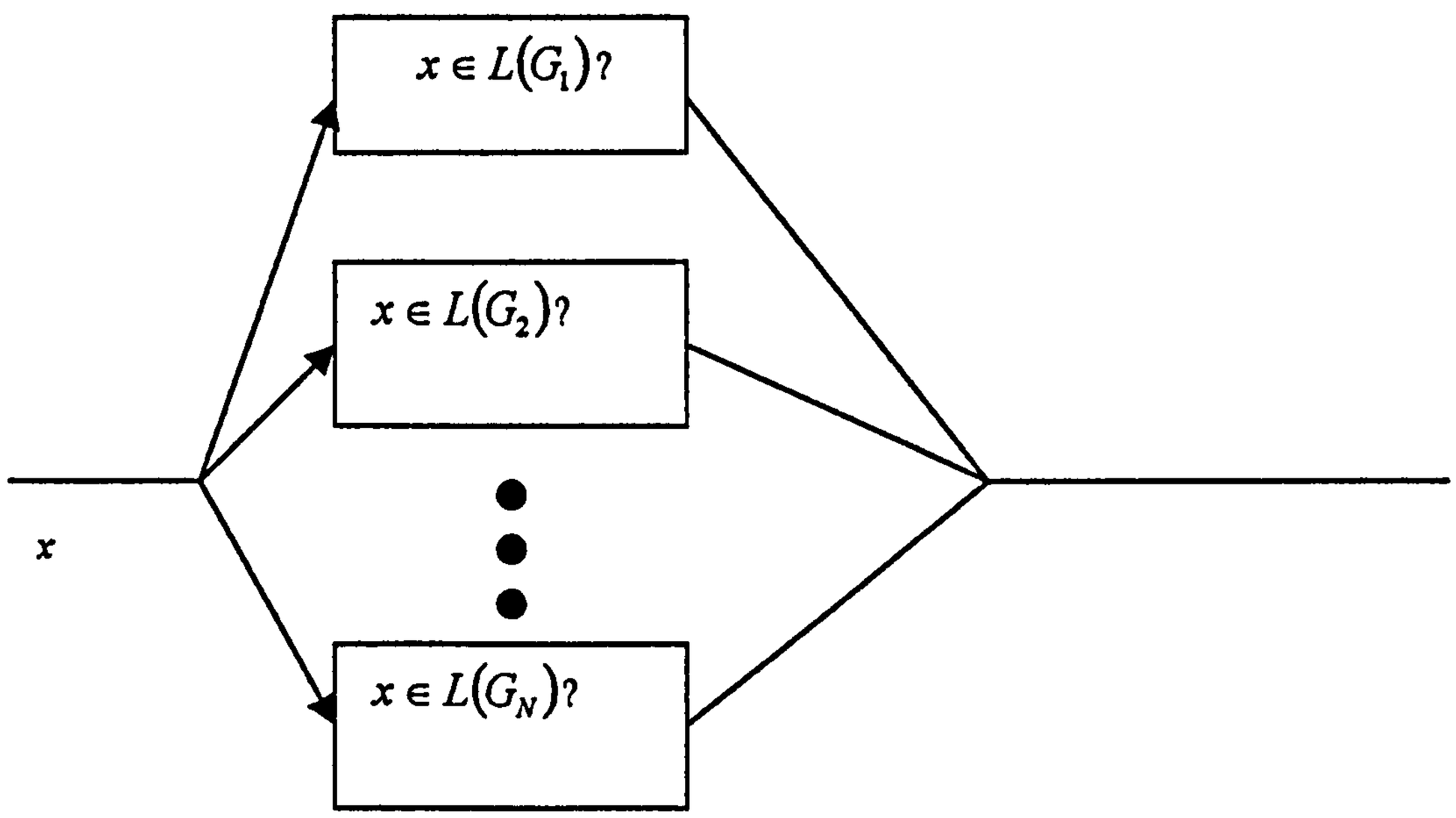


Figure 2.5 Block diagram for syntactic pattern recognition

Parsing is the process of determining whether a word is in the alphabet. There are two principal methods of parsing: namely top-down and bottom-up. With top-down parsing the derivation tree is constructed from the root to the leaves and from the leaves to the root with bottom-up parsing.

The above syntactic analysis assumes that the patterns are noise free and undistorted. To cope with these non-ideal cases the basic strategy can be augmented.

If knowledge of the possible distortions is known, it is possible to extend the ideal grammar to include error production rules. So the language represents not only the possible ideal patterns but also the possible distorted patterns.

It is possible that more than one grammar can generate the same pattern, so $x \in L(G_i)$ and $x \in L(G_j)$ where $i \neq j$. The basic grammar can be extended to include probability information; this type of grammar is called a stochastic grammar. Stochastic grammars are defined [Bunke 1990b] by

$$G_s = (N, T, P_s, S) \qquad 2.25$$

The terms N, T, and S are defined by Equation 2.19 and P_s represents the stochastic production rules.

If a pattern x does not match any available language, classification can be based on the closeness of the pattern to a language. Error correcting parsers can be used to try to reconstruct the pattern into the patterns determined by the available grammars in the classifier. The language that is the *closest* to the input pattern assigns itself to the input pattern. A distance measure called the one-dimensional weighted Levenshtein distance (1WLD) [Tanaka 1990] can be used to determine the closeness of the pattern to the language, $D(L(G), x)$. When correcting the pattern the operations of substitution, insertion, and deletion are used.

Structural recognition directly represents the input pattern and prototypes by suitable data structures; compare the input pattern with the prototypes and select the closest prototype

[Bunke 1990 b]. An example of expressing a pattern structure (house) as a tree structure using five primitives is given in [Sanfeliu 1990].

Grammatical inference is the process of teaching the classifier the syntactic structures. This can be defined as: the automatic learning of formal grammars from finite subsets of the language they generate [Miclet 1990].

Regular grammars can be inferred using techniques that consider the model as a finite-state machine or finite automaton. Inference methods include the successor method, Pao-Carr method, uv^kw algorithm, k-tails method, and tail-clustering method. The sequential machine is a slightly more complex model than the finite automaton. Stochastic grammars can be analysed as stochastic finite automata, although hidden Markov models have proved to be the most successful model.

Using these techniques in practice causes problems; real patterns are not context-free or regular so the theory represents an ideal situation, and inferring grammars from real patterns is very difficult. For these reasons, the grammars are manually designed for syntactic pattern recognition systems [Miclet 1990].

A different approach to rule learning has been investigated [Andre 1994], that uses Genetic Programming. Genetic programming was developed by Koza [Koza 1992] as a method of improving on Genetic Algorithms. Genetic Programming was originally developed using the programming language LISP, with the trees called S-expressions. LISP was chosen primarily because the S-expressions can be easily manipulated as data. The S-expressions use prefix notation giving a clear equivalence with a tree. Other computer languages can be used to represent the trees, for example C++ [Grimes 1995].

Genetic Programming was used to recognise the letter C from a recognition set of various characters in various fonts. Each character was pre-processed to extract the boundary, whose co-ordinates are placed in a linked list. Further processing is carried out to repair holes in the boundary that are caused by noise. The outer boundary is split into four segments to provide the metrics for the structural recognition process. An example of this segmentation for the letter C is shown in Figure 2.6.

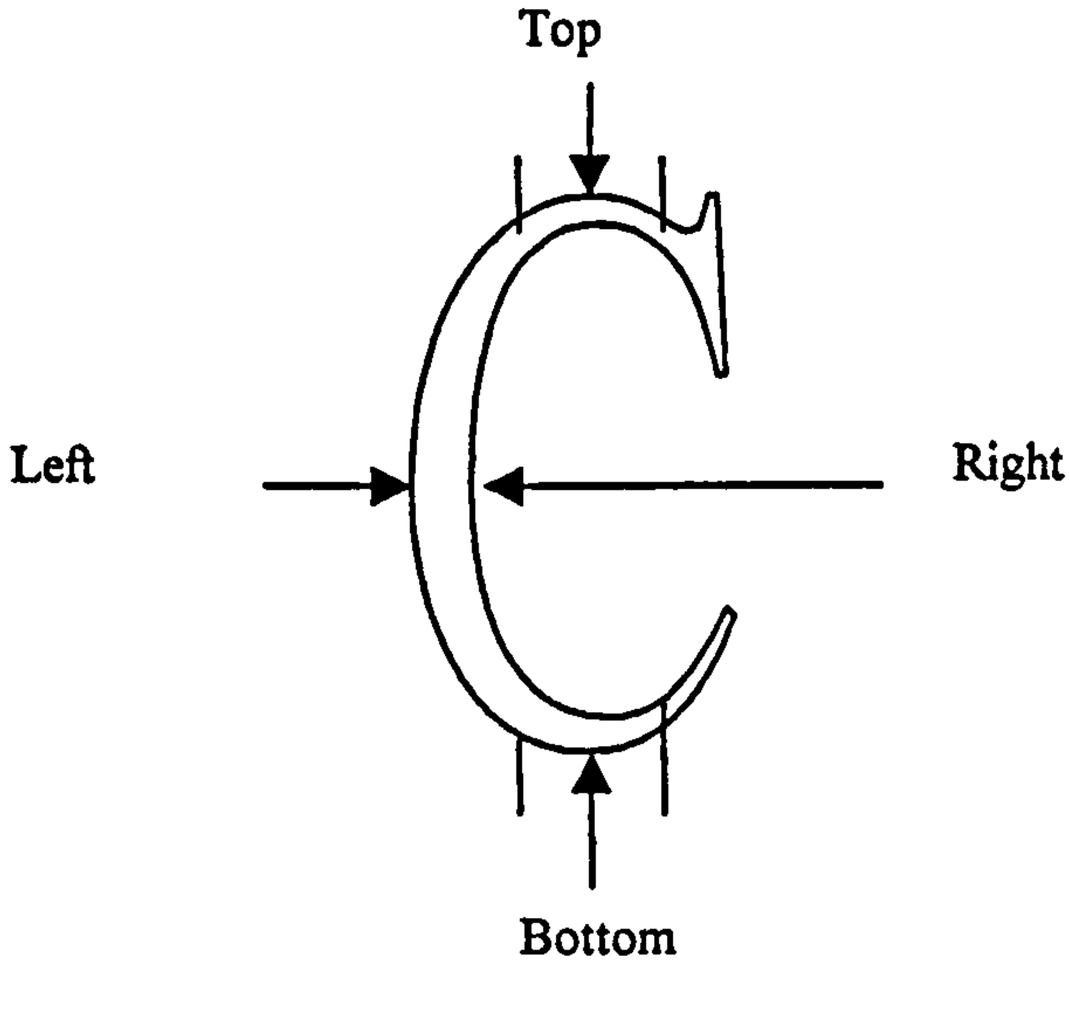


Figure 2.6 Segmentation of the letter C

After segmenting the outer boundary into the four segments called top, left, right, and bottom. The bounding box values (maximum and minimum row and column) are calculated for each hole, each segment, and the whole character. The number of pixels in each hole boundary is determined, and finally the segments are ranked in order according to their number of pixels.

Syntactic and structural techniques have been used for several computer vision applications, for example, three dimensional object recognition, Chinese character recognition and general purpose recognition.

Three dimensional object recognition

Attributed graphs have been used for three dimensional object recognition [Wong 1990]. Based on six generalised junction classes, 13 junction types were defined to represent the primitives from which the object is constructed. Constraints were put on the neighbouring node types to ensure that the object was valid. By matching six nodes, the camera-to-object transform was determined and the 3-D object was matched onto a 2-D plane to determine the other nodes.

Chinese character recognition

Chinese characters can be divided into four levels: whole character level, parts of character level, stroke level, and line segments level. Analysis of the Chinese language gives 7,000 - 10,000 whole characters, 200 - 500 parts of character level, 40 - 80 strokes, and 4 - 8 straight-line segments. Classifying Chinese characters can therefore be challenging. Eight angular relationships between primitives were defined. The spatial relationship between primitive A and B was defined by;

$$REL(A,B) = \left(E, re^{j\theta}\right) \qquad 2.26$$

This is shown in Figure 2.7.

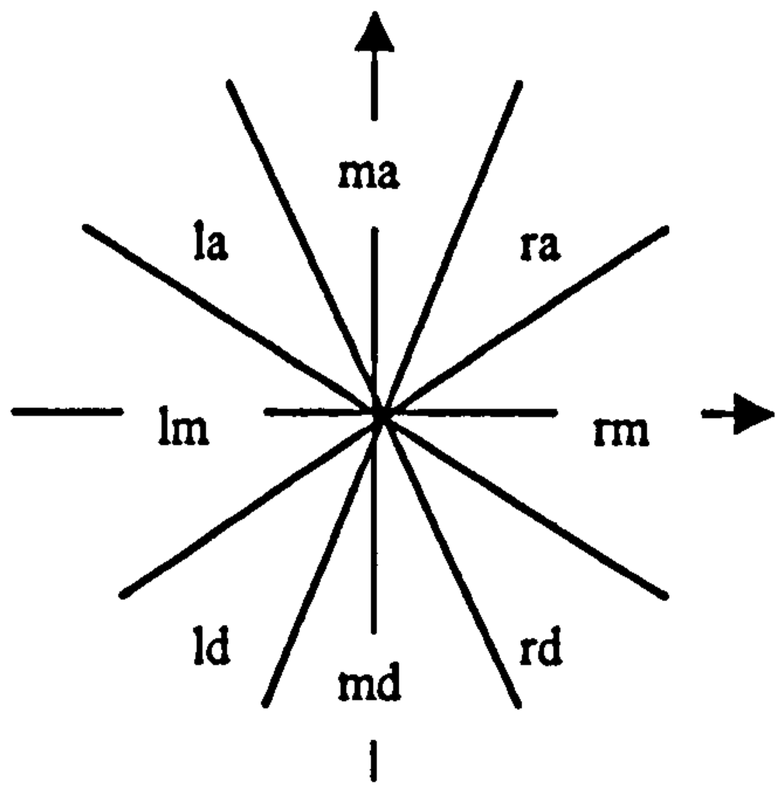


Figure 2.7 Spatial relationship primitives

Using an attributed grammar (containing both syntactic and semantic components), on-line Chinese character recognition has been achieved [Tai and Liu 1990].

General purpose recognition

An on-line general purpose line drawing analysis system, called MIRABELLE [Mohr 1990], has been developed. Both straight line (L) and circular arc (C) primitives were used. The head of a primitive was denoted by the O symbol and the tail by the X symbol.

Primitives are grouped into classes according to the angle that a line joining the head to the tail makes with the horizontal, and the angular tolerance. As an example, the primitive L(45, 35) denotes a class of line segments which the fall within the boundary of 45 ± 35 degrees with the horizontal. These classes are normally given names by specifying the class name for the class definition, for example;

horizontal: L(0, 10)

These sub-patterns can be assembled into more complex patterns using five assembly operators; +, x, o, *, and -. The tail of the resulting pattern is always the tail of the second operand, and similarly the head of the resulting pattern is always the head of the first operand. The - operator is a unary operator and permutes the head and tail, the other operator are binary operators.

Defining primitives v, h, and c, examples of these operators are shown in Figure 2.8.

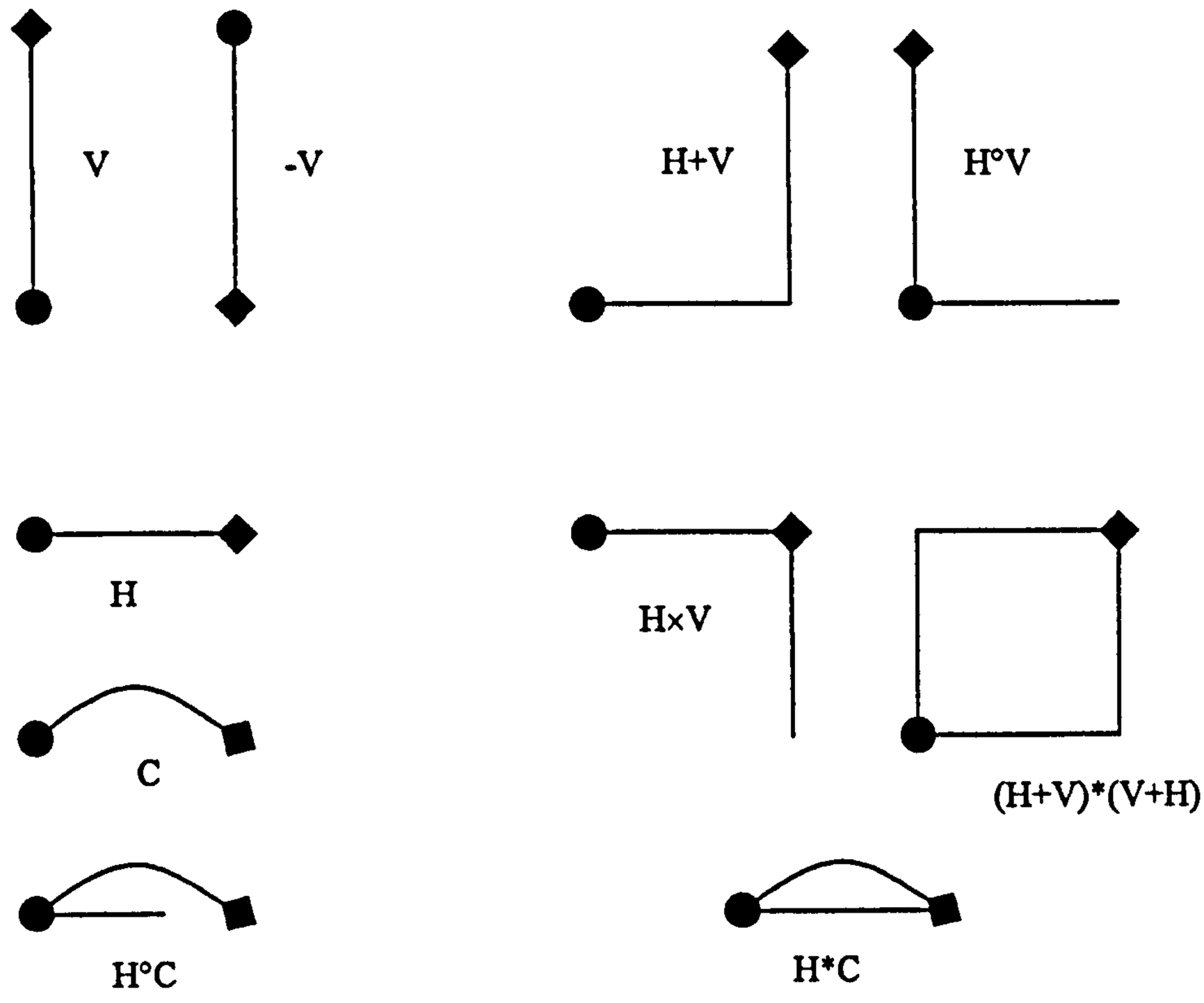


Figure 2.8 Example of the five operators

In addition to the assembly operators, 17 topographical operators were defined to allow imprecise assembly based on the topographical relationship of the sub-shapes.

MIRABELLE was able to recognise hand-drawn sketches, with a recognition time in the region of one to two seconds.

In this research, features will be assigned (see Chapter 5) after the network has learnt the relationships.

An invariant pattern recognition vision system has been proposed [Shepherd et al. 1992] that is capable of shift, rotation and scale invariant pattern recognition. The images are smoothed to remove noise, the edge image is broken down into a number of non-bifurcating contours, the contours are separated into linear and non-linear segments and described by metrics, and finally encoded into invariant metrics. Comparison is carried out by comparing the invariant representations. A flowchart for this system is shown in Figure 2.9.

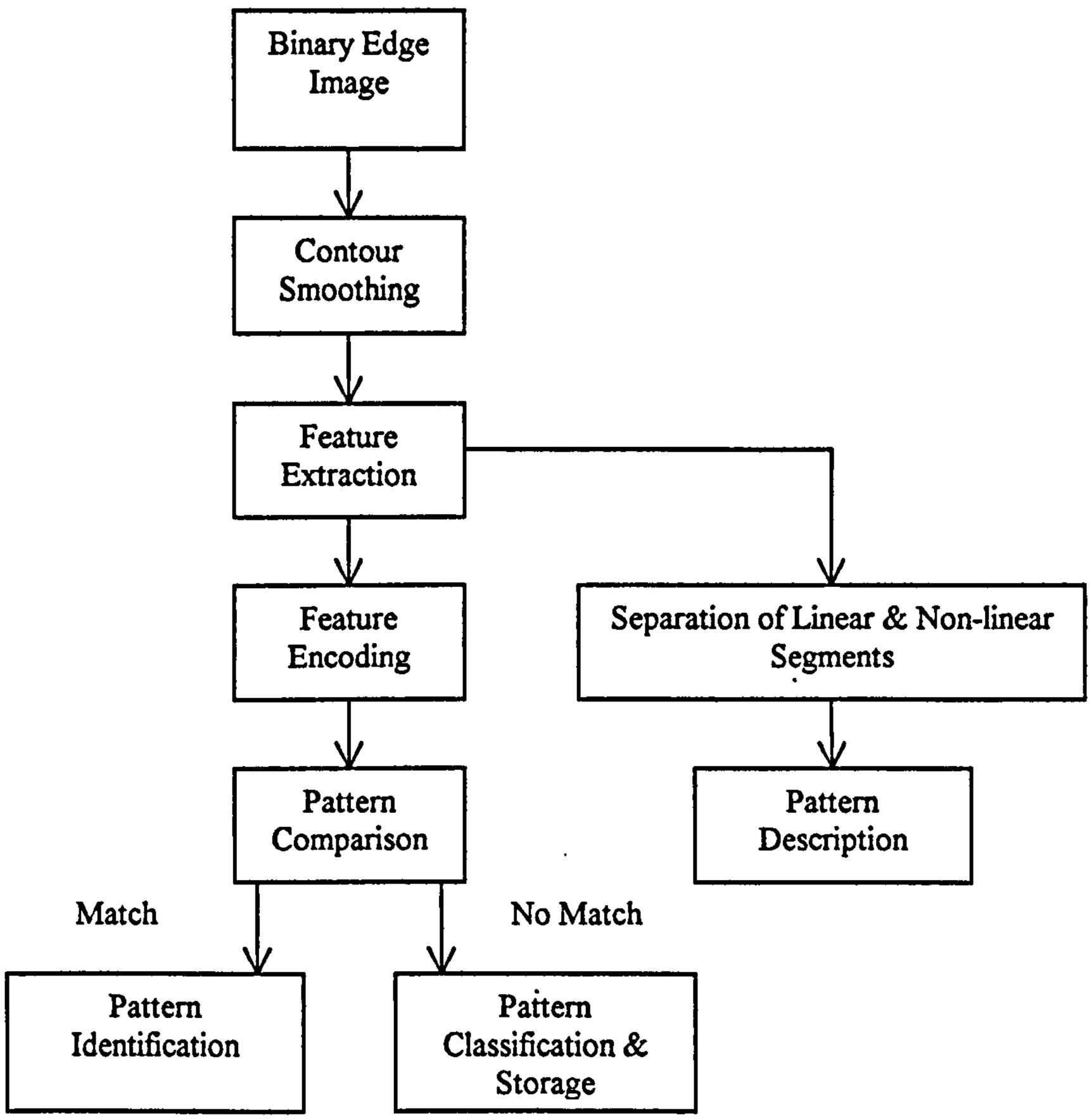


Figure 2.9 Flowchart for structural vision classification

Although this method identifies the low-level features and determines invariant metrics between the features that are used for pattern classification, the detailed structural relationships between these features are not determined.

2.4 NEURAL NETWORKS

The majority of today's computers are based on the Single Instruction Stream Single Data Stream (SISD) architecture developed by John Von Neumann. In 1946, he developed a set of principles that define a computer. These can be summarised as follows [Harvey 1971]:

- Instructions are coded as numerical digits and are stored in the machine.
- Instructions and data are stored together as words and the machine makes no distinction between them.
- Numbers are stored and manipulated in binary form.

Other contemporary researchers looked to biological systems for inspiration. This line of research lead to the investigation of neural behaviour and the connectivity of neurons in brains, in the hope of developing computers which behaved in a similar manner to brains. This research lead to the field of neural networks (sometimes known as artificial neural networks) which became a branch of Artificial Intelligence. Interest in neural networks began to decline towards the end of the 1960s. However there was resurgence in the 1980s, due to the progress being made in neural networks, coupled with the slow progress of other Artificial Intelligence methods.

Computers are very good at numerical computations, and repetitive tasks, and very poor at vision, speech recognition, and reasoning, whereas brains are very good at vision, speech recognition, and reasoning and very poor at numerical computations and repetitive tasks. It was considered that using some techniques that are based on the architecture of brains could lead to the development of computers with the best of both systems.

From the description for a neuron given earlier, simple principles of operation can be deduced;

- A neuron has many inputs (dendrites) and one output (axon).
- Neurons connect with each other at connections called synapses.
- Some synapses excite the neuron (excitatory), whereas other synapses inhibit the neuron (inhibitory).
- The cell body *sums* the input signals and fires when a threshold is reached.
- Signal levels are expressed by the number of polarity spikes/second.
- The output reaches a saturation level.

A mathematical model of such a neuron can be greatly simplified by expressing the signal level in spikes/second as an analogue level.

2.4.1 MCCULLOCH-PITTS NEURON

McCulloch and Pitts [McCulloch 1943] developed the first example of a mathematical model for a neuron. This model has a parameter, known as the *weight,* that represents the synaptic strength. The individual contribution that a signal from a preceding neuron or sensor makes is the product of the signal and the weight. This neuron had both excitatory and inhibitory synapses and is shown is Figure 2.10.

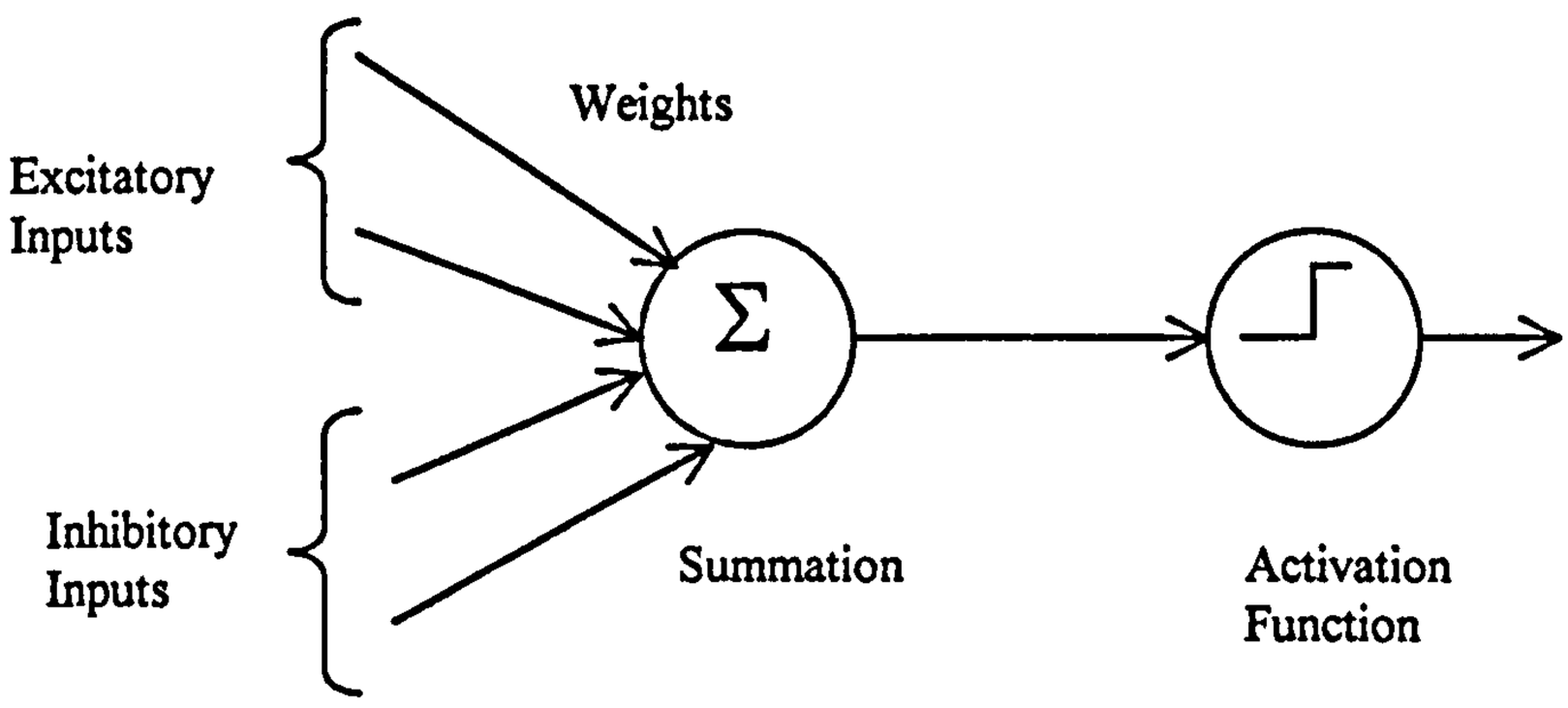


Figure 2.10 McCulloch-Pitts neuron

The neuron is normally described as having binary input levels (0 or 1). Firing takes place when there is input to inhibitory synapses and the sum of the input to the excitatory synapses has reached a threshold. With a suitable threshold and inputs wiring, the neuron could perform logic tasks such as AND, OR, and NOT. However the neurons could not be trained and they were not able to generalise.

Modelling the saturation of the output function is carried out by an activation (or squashing function), a number of different examples is shown in Figure 2.11. This figure shows the following types of activation functions:

A Bipolar threshold logic

B Hard limiter

C Unipolar threshold logic

D Bipolar sigmoidal

E Unipolar sigmoidal

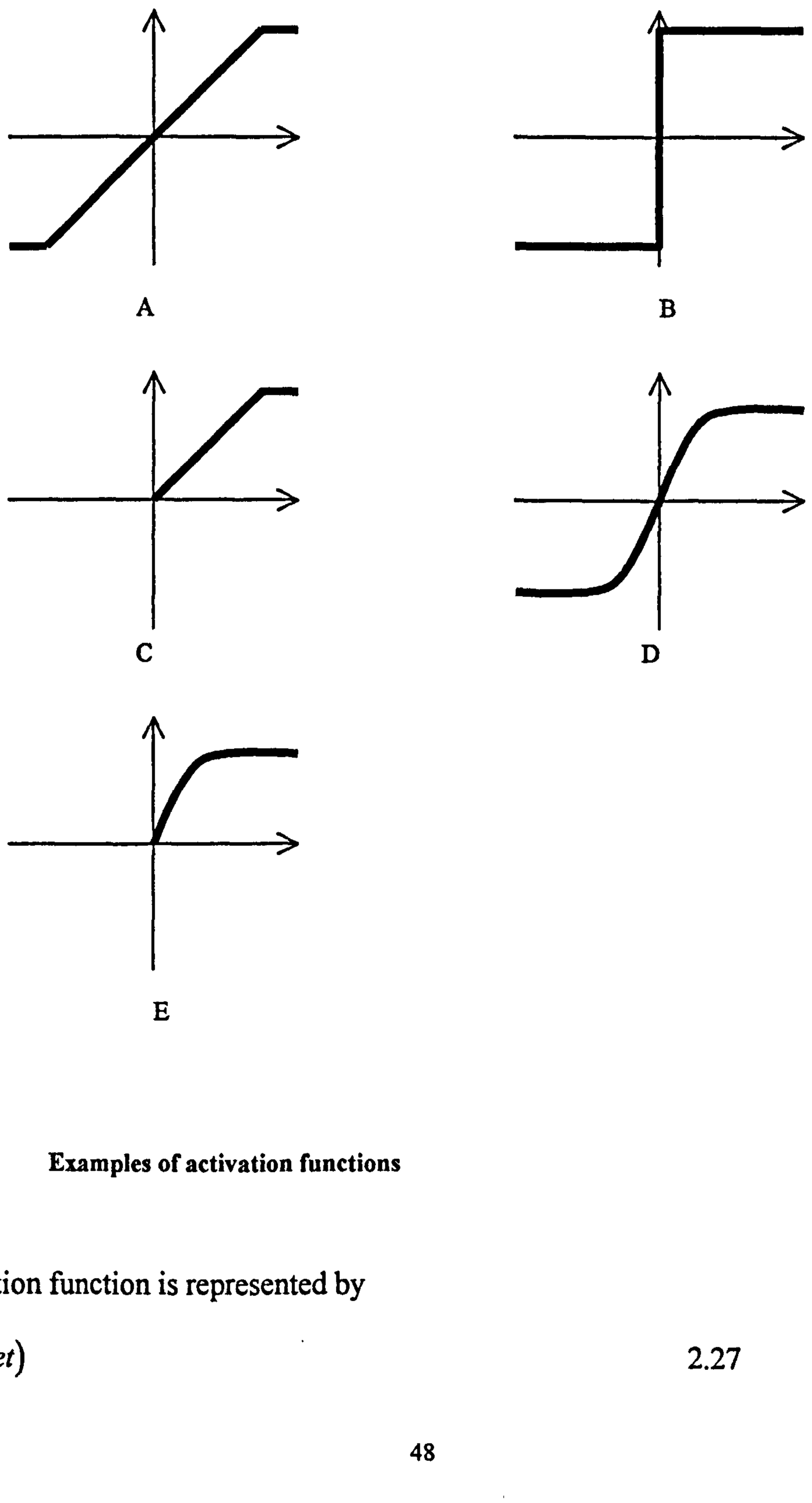


Figure 2.11 Examples of activation functions

The activation function is represented by

$$out = f(net) \qquad 2.27$$

Different forms of activation functions were used in this research.

The hard limiter was used in the original McCulloch-Pitts neuron, and neurons based on this type of neuron became known as Perceptrons.

Neural networks were difficult to implement. One early attempt to simulate a McCulloch-Pitts neuron was undertaken by Frank Rosenblatt [Rosenblatt 1958, 1959]. This electronic system used motor-driven potentiometers to provide the weights and their means of adaptation. The inputs were provided using photoreceptors, and the system was used to classify shapes that had been drawn on a grid, giving a crude raster image.

Neural network training can be classified into two groups [Wasserman 1989]: supervised and unsupervised.

Supervised training requires knowledge about the desired fully trained state of the system. Knowing the input to be presented, and the desired output, a training pair can be constructed. Comparing the actual output from the neurons with the desired output, an error can be determined. This error can be used to adjust the weights, and train the system. The training continues until the error is acceptable low, or some other criterion has been reached. Although this model has been successful, it is not seen as a plausible training method in biological systems. Human beings are able to learn a variety of complex tasks, for example vision, without being trained by an external supervisor.

Unsupervised training is a more biologically plausible method of training. It does not require a training set. Training involves the presentation of input vectors, and the network is able to determine how to adjust the weights to produce output vectors that are consistent

with the various classes of inputs. Unsupervised training was initially developed by Teuvo Kohonen [Kohonen 1984]. As the network determines how to assign the weights, there is generally no way of knowing which output layer neuron will respond to which feature. The network input-output relationships have to be determined after training.

2.4.2 SINGLE LAYER PERCEPTRON

A single layer of perceptrons, which is unsurprisingly called a single layer perceptron, is taught by supervised learning. Its network architecture is shown in Figure 2.12.

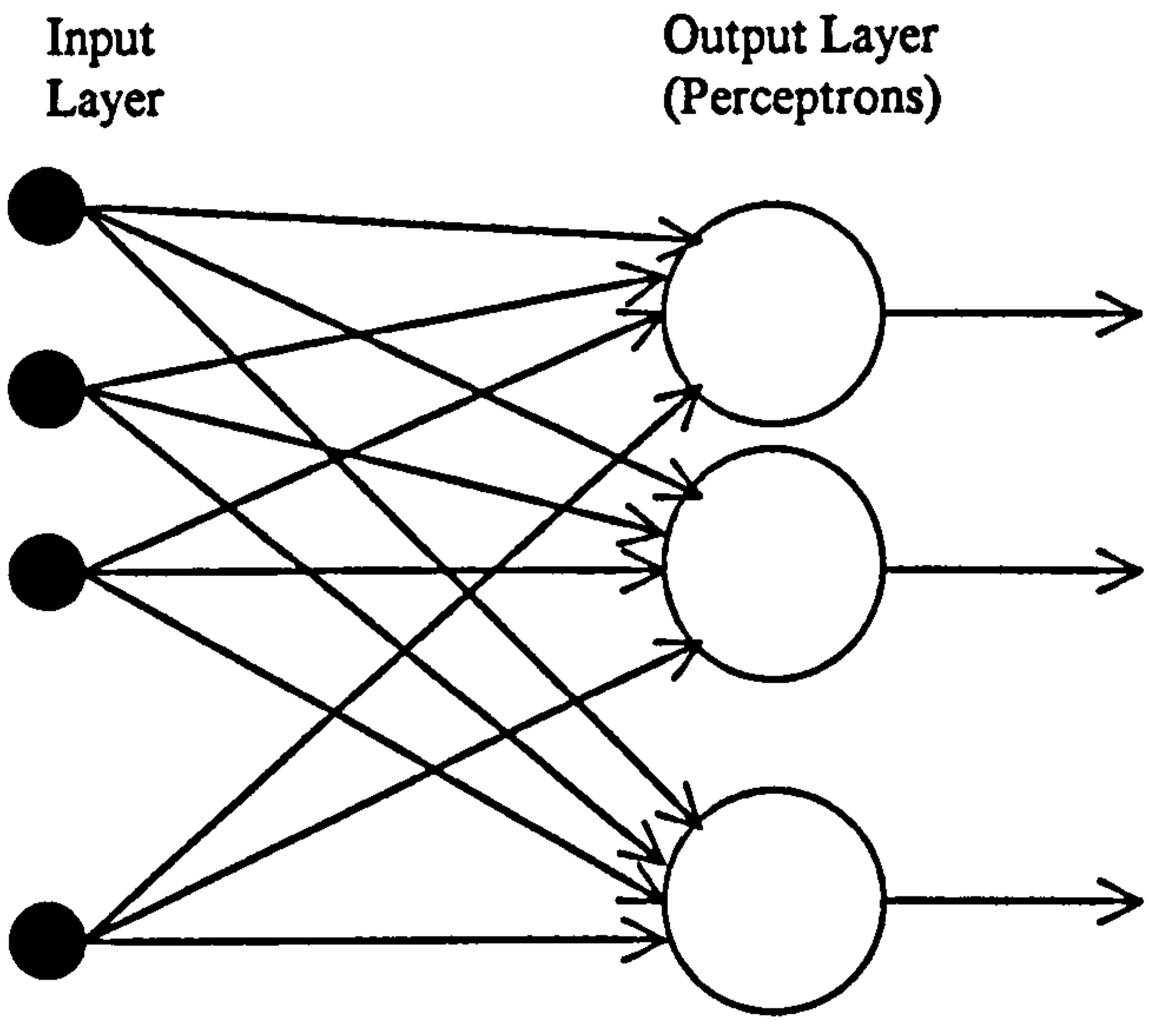


Figure 2.12 Single layer perceptron

The total activity reaching the cell body is given by;

$$net = \sum_{i=1}^{n} w_i x_i + w_0 \qquad 2.28$$

The parameter w_0 is an offset or bias and represents the firing threshold, and introduces a fictitious input (x_0) for w_0 which has a numerical value of +1. Equation 2.28 can be simplified, giving

$$net = \sum_{i=0}^{n} w_0 x_i \qquad 2.29$$

The perceptron is taught using supervised learning. It was mentioned earlier that with supervised learning there is an input-output training pair that allows the error between the actual and desired outputs to be determined. For each neuron, the error could be either positive or negative. One method of training the neuron is to minimise the mean of the square of the error for all training patterns. Only considering the summing part of the neuron, one way of expressing the error for presentation p is

$$error_p = d_p - net_p \qquad 2.30$$

The quantity that should be minimised is

$$E = \sum_{p=1}^{P} \frac{\left(d_p - net_p\right)^2}{P} \qquad 2.31$$

One method of minimising the error is to find the gradient of the error and move (the weights) a small distance in the direction of the gradient. The gradient is further evaluated and the weights are adjusted. This continues until the error has reached an acceptable limit or another stopping criterion has been reached.

For a network with n inputs and m neurons, the actual network output for neuron j can be represented by out_j and the desired output by d_j. The weight for synapse i and neuron j is given by w_{ij}, and the input to synapse i is given by x_i. The error is given by subtracting the actual output from the desired output. Training is accomplished in discrete steps,

denoted by k. The weight adaptation rule is given by [Widrow and Hoff 1960, Lippmann 1987];

$$w_{ij}(k+1) = w_{ij}(k) + \eta\left(d_j(k) - out_j(k)\right)x_i(k) \qquad 2.32$$

This is also known as the delta rule. The correction is proportional to the error and the input, with a constant of proportionality (η) known as the learning rate.

Initially the weights are set to small random values. The gain term, η, has a value ranging from 0.0 to 1.0. The neuron outputs are calculated using equations 2.27 and 2.29, and the weights are adjusted using equation 2.32. Training the network is accomplished using the following algorithm:

initialise weights to small random values

counter = 0

REPEAT

counter = counter + 1

present input vector for loop number counter

adapt weights using desired output for loop number counter

UNTIL exit condition has been reached

The exit condition can be a fixed number of loops or the errors reaching an acceptably low level.

Perceptrons normally have a hard limiter as the activation function, so each neuron will give a clear Boolean output. The boundary between the decisions is where the value for *net*

is 0. Considering equation 2.28 for a two input neuron and setting the output to 0 and rearranging gives

$$x_2 = -x_1\left(\frac{w_1}{w_2}\right) + \frac{w_0}{w_2} \qquad 2.33$$

Extending this analysis to a neuron with three synapse inputs, the decision is a plane that separates the two classes of output. Four or more synapses give a decision surface that is a hyperplane with dimensionality equal to the number of synapses. As the decision surface is a hyperplane, the single layer perceptron is able only to classify problems which are linearly separable.

Single layer perceptrons were extensively analysed by Minsky [Minsky 1969]. This analysis showed that they were incapable of learning simple tasks, including the exclusive OR function. This book resulted in a number of researchers leaving the field of artificial neural networks, because at that time no techniques were known for training multilayer perceptrons.

2.4.3 ADALINE

The ADALINE is a neuron that is very similar in construction to a perceptron; ADALINE is an acronym for ADAptive LInear NEuron or ADAptive LINear Element. It was developed by Widrow [Widrow 1962], and has a construction shown in Figure 2.13.

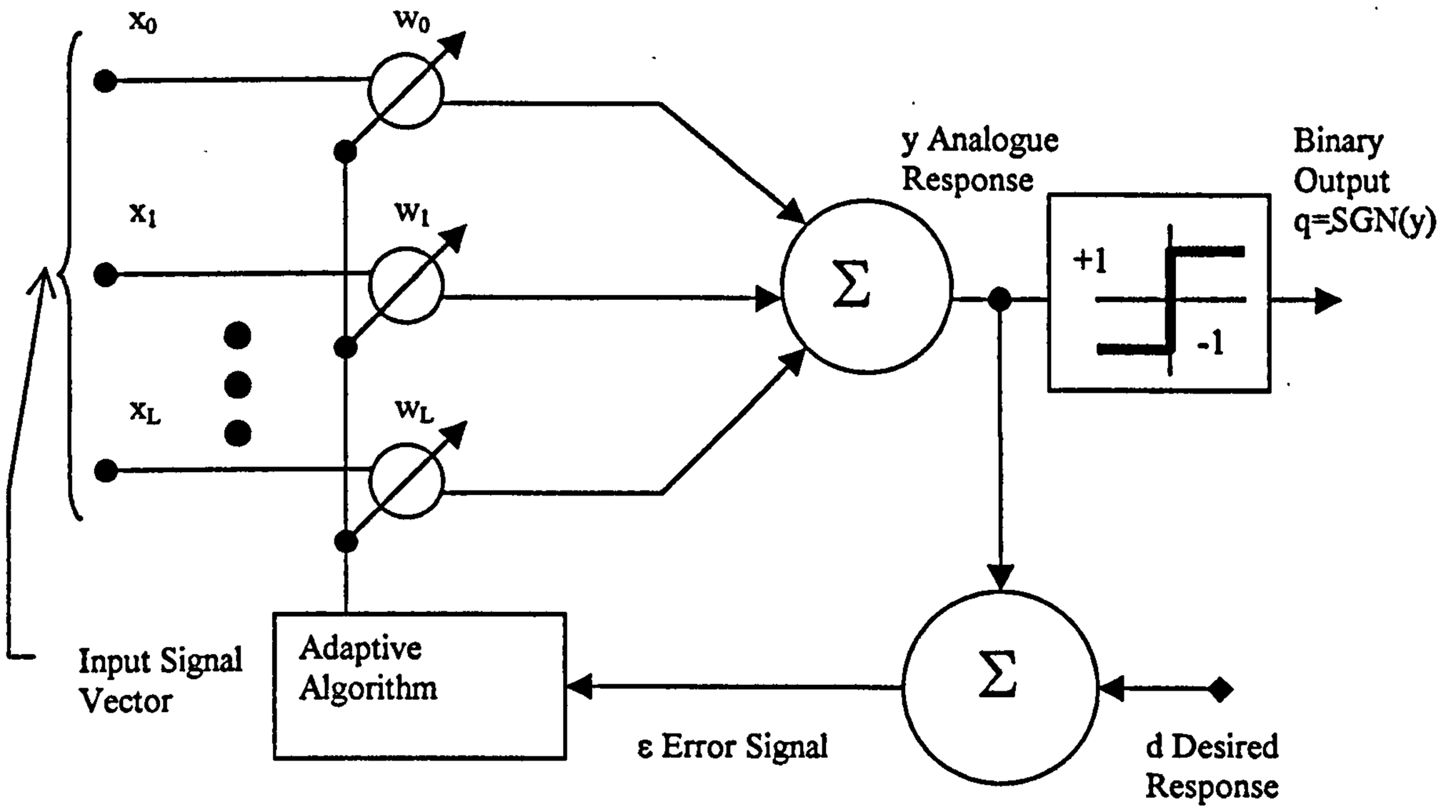


Figure 2.13 A single ADALINE

Teaching the ADALINE is very similar to teaching the perceptron. The weight adaptation algorithm is slightly different, and is more in keeping with derivation of the learning algorithm where the error is calculated by considering the desired output and the output from the summing unit.

From 2.32

$$w_{ij}(k+1) = w_{ij}(k) + \eta\left(d_j(k) - net_j(k)\right)\ x_i(k) \qquad 2.34$$

With a suitable choice for the offset weight, w_0, the ADALINE can be made to emulate logic gates, for example AND, OR and MAJority (fires when a majority of inputs are active) logic gates.

Along with the perceptron, the ADALINE suffers from linear separability problems. An attempt was made to get around this problem using more than one ADALINE arranged in parallel and logic gates in an arrangement called Madaline (many ADALINEs).

An invariant vision system using ADALINEs has been proposed [Widrow and Winter 1988]. The building block of the system is a slab of ADALINES, with the output from each ADALINE going to a MAJ ADALINE, giving one output for the whole slab. The weights of the ADALINES were chosen so that the slab output will be insensitive to translation. There are several of these slabs, with each one having a different set of weights. When an image is presented to the network, each slab will produce an output that is independent of the position of the object in the image.

Presenting different images will evoke a pattern of responses from the slabs. This pattern can be decoded by a two layer ADALINE adaptive descrambler. This is shown in Figure 2.14.

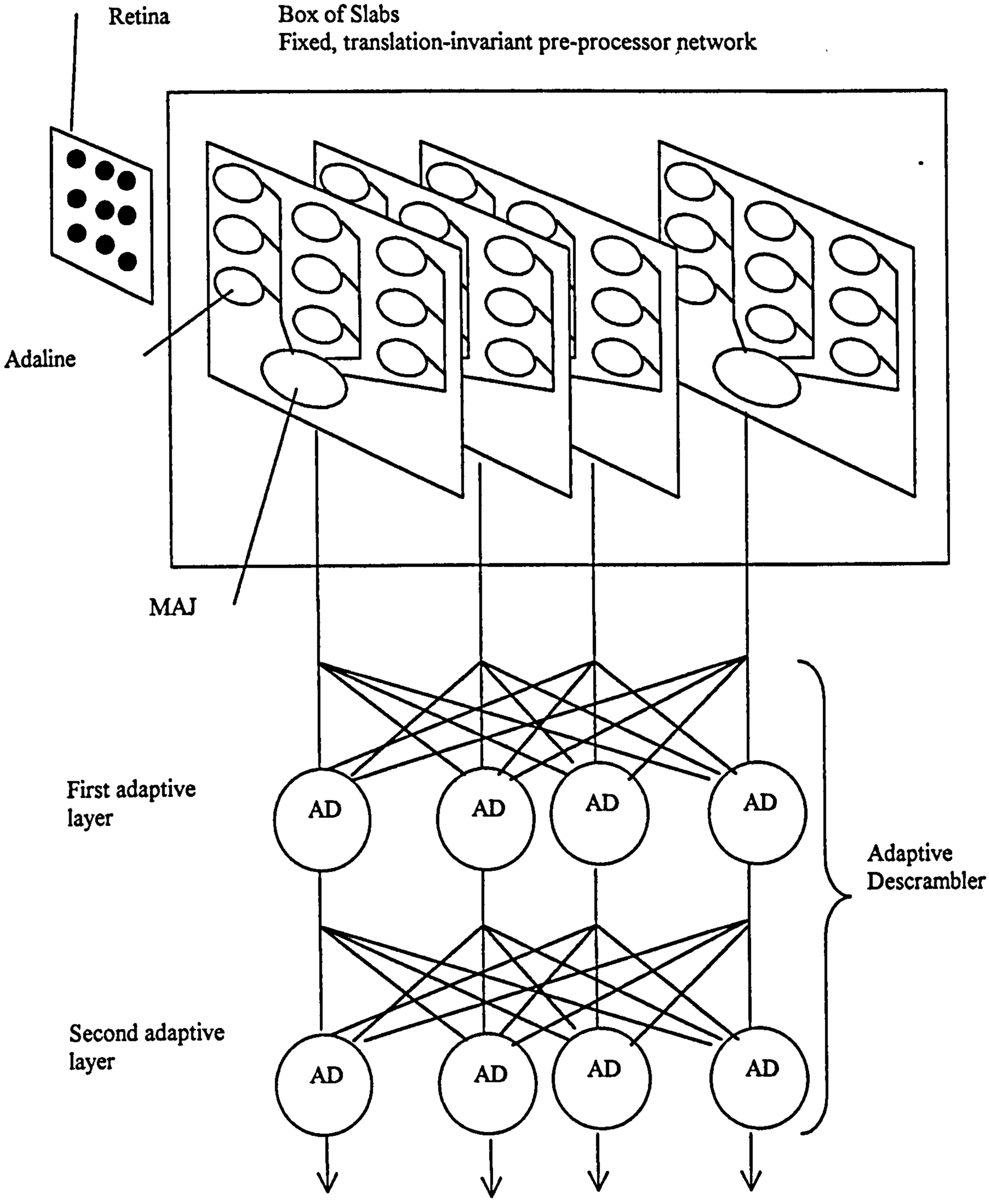


Figure 2.14 Translation invariant ADALINE classifier

Each ADALINE has inputs that cover the whole of the retina. Assuming that the top left hand ADALINE has a two-dimensional array of weights (W_1), the ADALINE directly below the top left hand ADALINE has an array of weights $T_{D1}(W_1)$, where T_{D1} represents

the *translate down one* operator. This operator shifts the neurons down by one, and the weights that fall off the end at the bottom are wrapped around to the top. Generalising this operator to T_{Dy}, so that $T_{Dy}(W_1)$ represents the weights of the ADALINE y below the top left hand ADALINE, the weights in any position can be represented.

Similarly the weights can be shifted and wrapped in the right direction, a combined down-right shift operator can be designated by $T_{Rx}T_{Dy}$. Considering a 4 by 4 retina, the weights for the ADALINEs in a slab would have the following representation:

(W_1)	$T_{R1}(W_1)$	$T_{R2}(W_1)$	$T_{R3}(W_1)$
$T_{D1}(W_1)$	$T_{R1}T_{D1}(W_1)$	$T_{R2}T_{D1}(W_1)$	$T_{R3}T_{D1}(W_1)$
$T_{D2}(W_1)$	$T_{R1}T_{D2}(W_1)$	$T_{R2}T_{D2}(W_1)$	$T_{R3}T_{D2}(W_1)$
$T_{D3}(W_1)$	$T_{R1}T_{D3}(W_1)$	$T_{R2}T_{D3}(W_1)$	$T_{R3}T_{D3}(W_1)$

The weights for the top left hand ADALINE in each slab would be chosen randomly.

A similar system can be used to give rotational invariance to the network. Restricting the network to four angles of rotation (0, 90, 180, and 270 degrees clockwise), the above system can be extended using the rotation operator R_{Ca}, where *a* represents 90 degrees of clockwise rotation. Four times as many slabs are required, one set for each 90 degrees of rotation. As an example, the weights for the 90 degrees clockwise slab is given below;

$R_{C1}(W_1)$	$T_{R1}R_{C1}(W_1)$	$T_{R2}R_{C1}(W_1)$	$T_{R3}R_{C1}(W_1)$
$T_{D1}R_{C1}(W_1)$	$T_{R1}T_{D1}R_{C1}(W_1)$	$T_{R2}T_{D1}R_{C1}(W_1)$	$T_{R3}T_{D1}R_{C1}(W_1)$
$T_{D2}R_{C1}(W_1)$	$T_{R1}T_{D2}R_{C1}(W_1)$	$T_{R2}T_{D2}R_{C1}(W_1)$	$T_{R3}T_{D2}R_{C1}(W_1)$
$T_{D3}R_{C1}(W_1)$	$T_{R1}T_{D3}R_{C1}(W_1)$	$T_{R2}T_{D3}R_{C1}(W_1)$	$T_{R3}T_{D3}R_{C1}(W_1)$

Using similar reasoning the system can be expanded to cope with scale invariance. By defining a point of expansion, the positions of the weights can be scaled, with the magnitude of the weights scaled in inverse proportion to the square of the linear dimension of the retinal pattern.

A variant of this device has been used for coin recognition [Fukumi et al. 1992]. A rotation invariant network was constructed in a similar manner to above, although the arrangement of neuron inputs in each slab is in the form of a circular array instead of the more usual rectangular array. The neuron inputs are in the form of sectors of concentric circles. Using a circular array is valid because only the classification of circular coins is required. An image is scanned, a pre-processor determines the centre and diameter of the coin, the image is scaled to the diameter of the circular array, and the centre of the coin is aligned with the centre of the array.

The neurons in each slab are neurons with a sigmoidal activation function, and the slab output is provided by an approximate majority vote taker that also has a sigmoidal output. Classification is undertaken by a trainable multilevel network. This network is called a-CONE, or analogue coupled neuron, that behaves in a similar manner to backpropagation.

An adaptive de-scrambler using sigmoidal perceptrons has been used for rotational invariance [Tanomaru and Inubushi 1995].

The method proposed by Widrow and its variants, provide a practical method for rotation invariant classification. Features are not extracted and the structure of the object is not considered.

2.4.4 MULTILAYER PERCEPTRON

The breakthrough in developing multilayer perceptrons was the development of the learning algorithm called backpropagation. This method was originally developed by Werbos [Werbos 1974]. Unfortunately this was ignored and rediscovered by Parker [Parker 1982], and again ignored only to be rediscovered by Rumelhart, Hinton, and Williams [Rumelhart at al. 1986].

A single layer perceptron can only classify linearly separable problems. A multi-layer perceptron is able to classify a greater range of problems. Because the summation is linear, when two or more layers are combined, the combination of these layers reduces to a single layer. To prevent this from happening, the output from each layer must contain non-linear elements.

Backpropagation is a method of supervised learning in a multilayer perceptron by adjusting the synapse weights in a manner that will reduce the error between the desired and actual network output. This method relies on differential calculus, and involves the differentiation of the activation function. Using a hard limiter presents problems because it has an infinite gradient at the threshold. To get around this problem a sigmoidal (non-linear) function that has a simple derivative is normally used. The activation normally used is given in equation 2.35

$$out = f(net) = \frac{1}{1+e^{-net}} \qquad 2.35$$

Training is carried out using a similar training algorithm to that for the perceptron, although an extra loop is required to train the neurons in each layer. Multilayer perceptron

learning can be improved by modifying the delta rule to include a momentum term [Rumelhart et al. 1986].

A shift and scale invariant classifier has been implemented using the multilayer perceptron with backpropagation training [Elliman and Banks 1990]. This system uses a specialised neural network for positioning the object centroid at the centre of the retina. To assist in the translation, neurons are used which have fixed weights that vary with the length of the dendrite, with the longer the length the smaller the weight. The output from the layer is an image shifted towards the centre of the retina. This output image can be repeatedly fed back to the input of this layer, and when the layer has settled the object will be correctly positioned.

After aligning the image it is converted to radial polar co-ordinates and normalised for size by shifting values in the radial direction. The image has been converted into a shift and scale independent form.

The next layer of processing is feature extraction, which is accomplished using a Hough Transform. The accumulated scores are split into six ranges and four possible features can be detected. Giving the output from the accumulators in the form of a two-dimensional array. The multilayered perceptron classifies images using the output from this Hough transform.

It was found that the classification performance was limited by the coarse Hough array.

Another neural network that also shifts the image has been proposed [Lin and Wang 1996]. Following the translation layer is a rotation normalisation layer, utilising neurons with

Neocognitron like behaviour, followed by a feature extracting layer and two feed-forward layers.

Multilayer perceptrons have been used with moment invariants to produce a system invariant to translation, rotation and scale [Gonzaga and Costa 1996], with an edge detected contour sequence [McNeil and Sarkodie-Gyan 1995], with the Fourier Transform [Zhu et al. 1996], using simple affine transforms [Lee et al. 1995], using a wavelet transform [Sadovnik 1995], Radon Transform [Al-Shaykh and Doherty 1996] and DFT, fuzzy clustering and GA [Lee and Jang 1996].

2.4.5 KOHONEN'S SELF ORGANISING MAPS

Self-organised feature maps were developed by Kohonen to provide an unsupervised method of learning and to give the network a structure that would reflect the characteristics of the stimuli. The brain is organised into regions, and the neurons in each region will respond in a similar manner. Neurons in a layer can be connected laterally, where the output of one neuron can become an input to neurons in the same layer. For a *Mexican hat* profile of interactions, neighbouring neurons are excited by the activation of a neuron whereas more distant neurons are inhibited.

Using the above network configuration, an initial random assignment of weights can evolve so that the network will respond to specific input patterns by firing clusters or *bubbles* of neurons. However, the level of interactions involved can lead to a numerically intensive system, which affects its performance. The method developed by Kohonen [Kohonen 1988] can provide good self-organising results using a much simpler, and faster, system.

The neurons are arranged in a two-dimensional array with all the inputs connected to all the synapses. Similarly to other networks, the weights are assigned to small randomly generated values. An input pattern is applied to the network and for each neuron the difference between a vector of inputs and a vector of weights is determined. This difference is the Euclidean L2 distance norm.

All the neurons are in competition with each other, and the neuron with the smallest distance is chosen to be updated. Selecting neurons using a performance criterion and subsequently updating them is called competitive learning. Because this method considers a collective system of neuronal behaviour, the surrounding neurons are also updated. Numerous types of neighbourhood regions can be used for example circle, square, or hexagonal. The size of this region initially covers about half the layer and decreases monotonically with time.

The following algorithm can be used:

initialise weights to small random values

counter = 0

REPEAT

counter = counter + 1

present input vector for loop number counter

FOR all neurons **DO**

calculate distance squared = $\sum (x - w)^2$

END

determine neuron with smallest distance squared

determine the updating region around winning neuron

```
        FOR all neurons DO
                IF neuron is in region
                        adapt weights
                ENDIF
        END
UNTIL exit condition has been reached
```

Kohonen was dissatisfied with the original formulation of Hebb's Law, in that it only allowed the strength of the synapses to increase. The following learning rule (Kohonen learning) is used to update the weights:

$$w(k+1) = w(k) + a(k)(x(k) - w(k)) \qquad 2.36$$

$a(k)$ is a monotonically decreasing function with time.

It has been discovered that if the variances of each input component differ in order of magnitude, the self-organised maps can have an oblique orientation [Kangas 1990]. This can be corrected by having a weighting term for each Euclidean distance component, giving the weighted Euclidean distance given by equation 2.37.

$$d^2 = \sum \left(w^2 (E - m)^2 \right) \qquad 2.37$$

The values for the distance weights are calculated using a backward exponentially weighted average for parameter *d*.

2.4.6 ADAPTIVE RESONANCE THEORY

Adaptive Resonance Theory (ART) is a neural network that learns in an unsupervised manner. This network was developed to overcome the stability-plasticity dilemma, where the network must be stable to enable learnt patterns to be retained, and must be plastic to enable new patterns to be learnt [Carpenter and Grossberg 1987a, Carpenter and Grossberg 1988]. The network is sensitive to novelty and is capable of distinguishing between familiar and unfamiliar events, as well as between expected and unexpected results.

The original formulation of ART could only classify binary inputs and is known as ART1. Adaptive Resonance Theory has been further developed, for example ART2 [Carpenter and Grossberg 1987b] will work with non-binary input signals.

ART can be used for invariant computer vision using invariance filters such as Fourier-Mellon [Carpenter and Grossberg 1987c].

2.4.7 RADIAL BASIS FUNCTION

The radial basis function (RBF) is a neural network that consists of two layers [Broomhead and Lowe 1988, Park and Sandberg 1993, Zhao and Bao 1996], a hidden layer and an output layer. Neurons in the hidden layer give an output represented by the kernel function $K(\cdot)$, given by

$$Z_i(\mathbf{x}) = K\left(\frac{\|\mathbf{x}-\mathbf{c}_i\|}{d_i}\right) \qquad 2.38$$

This kernel function is typically a Gaussian function and the neurons have a localised response rather like the receptive field of a biological neuron.

The output from the output layer is given by the weighted sum of the outputs from the hidden layer.

$$y_k(\mathbf{x}) = \sum_{i=0}^{N} w_{ki} Z_i(\mathbf{x}) \qquad 2.39$$

Training of RBF is normally a two-stage process in which the hidden unit parameters are taught first followed by the final layer weights. This can be carried out in several ways, for example the orthogonal least squares method [Chen et al. 1991] and K-means algorithm [MacQueen 1967]. Using a Fourier Transform to perform shift invariance an RBF was used to classify radar targets [Zhau and Bao 1996].

2.4.8 HIGH-ORDER NEURAL NETWORKS

High order neural networks can be used for invariant pattern recognition [Perantonis and Lisboa 1992, Perantonis 1992]. For a retina with N pixels, each pixel i has position $r(i)$ and value s_i. A single layer network that considers the product of the input values can be defined. For the arrangement shown in Figure 2.15, the output of neuron number j of order p can be given by:

$$O_j(s) = f\left(\sum_{i_1} \sum_{i_2} \cdots \sum_{i_p} w_{i_1 i_2 \cdots i_p j} s_{i_1} s_{i_2} \cdots s_{i_p} \right) \qquad 2.40$$

The function $f(\ )$ is the activation function that is typically sigmoidal.

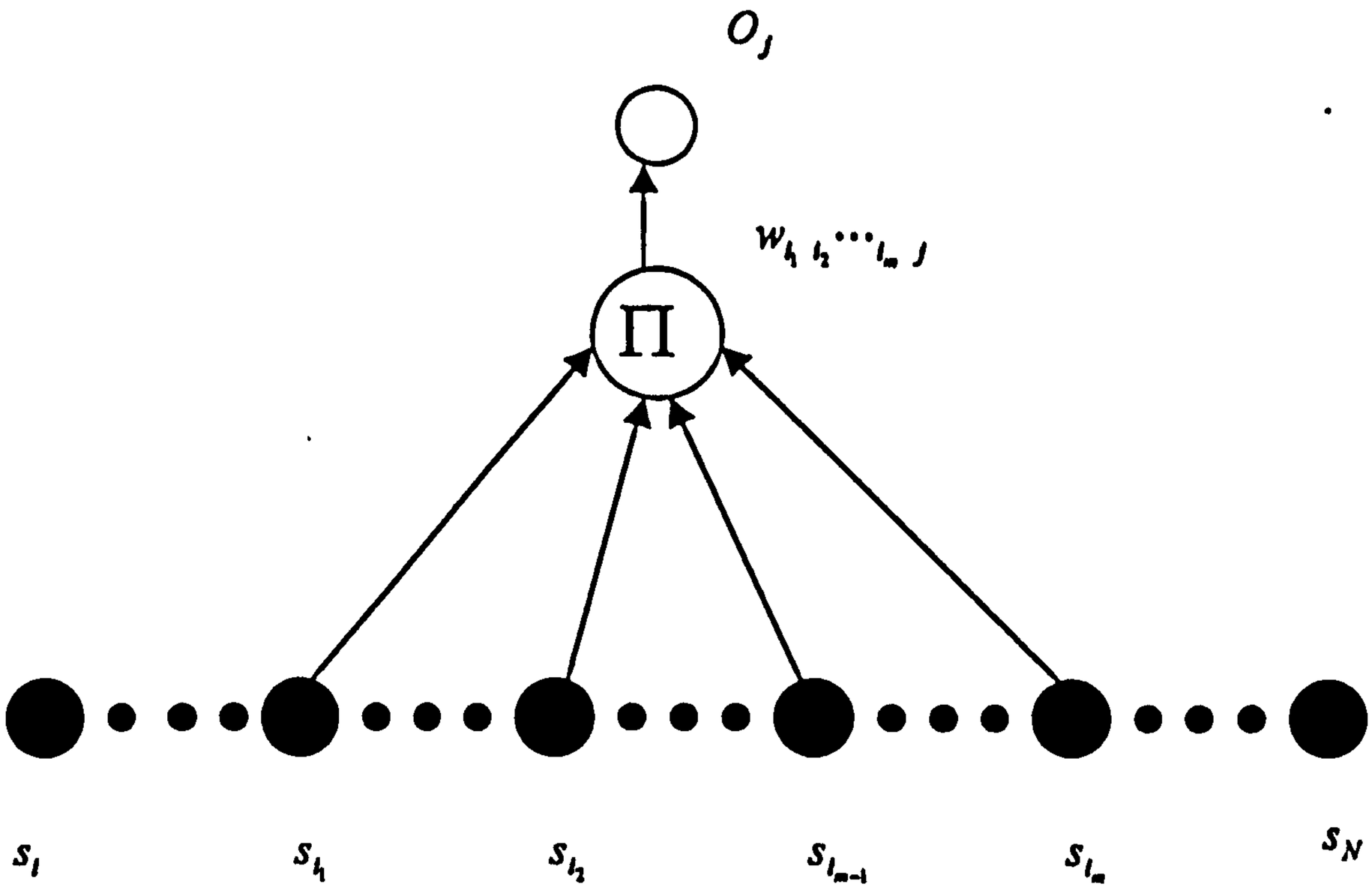


Figure 2.15 Connectivity of a high-order network

The weights are chosen from the geometry of the object and output will be invariant if the weights are chosen according to the equation;

$$w_{i_1 i_2 \cdots i_p j} = w_{k_1 k_2 \cdots k_p j} \qquad 2.41$$

For invariance to translation, rotation, and scale a third order network must be used.

High order neural networks have been applied to coarse coding [Kröner 1995], face recognition [Uwechue et al. 1995] and the topology has been optimised using Genetic Algorithms [Liatsis and Goulermas 1995]. The structure of the image is not considered.

2.4.9 ASSOCIATIVE MEMORY

Associative memory is one type of system that is capable of recalling the full recollection when only part of this stored memory has been presented to the system. The human brain has this feature. It is known that part of an experience such as a smell, a sound, or an image can recall the whole experience.

Associative memory has been used by Zhou and Chellappa to perform a number of visual processing tasks [Zhou and Chellappa 1992] such as;

- three dimensional static stereo vision
- three dimensional lateral motion stereo vision
- three dimensional longitudinal motion stereo vision
- computation of optical flow
- image restoration

Associative memory has been used in commercial applications, for example the WISARD system. WISARD is an acronym for WIlkie, Stonham, and Aleksander's Recognition Device [Aleksander and Stonham 1979, Austin and Stonham 1987, and Boyle and Thomas 1988]. This system can be used for face recognition, where the face does not have to be placed in exactly the same position and the system is tolerant to small changes in viewing angle. To produce this tolerance multiple images are associated with each face.

2.4.10 RESTRICTED COULOMB ENERGY

A network using Restricted Coulomb Energy (RCE) has been used for invariant pattern recognition [Li and Nasrabadi 1990]. This approach uses significant processing to extract the salient features from the image. After thresholding, the image is converted from Cartesian to radial-polar co-ordinates (origin at the centroid), scale normalising, splitting the object into 12 segments and extracting the following features:

- Boundary information
- Area information
- Maximum distance from origin
- Minimum distance from origin

The shift invariance capability is provided by using the centroid as the origin for the Cartesian to radial-polar co-ordinates transformation. The scale normalisation provides the scale invariance and the rotational invariance is provided by arranging the feature codes in a fixed order. The RCE net is similar in structure to the multilayer perceptron, having an input, a hidden and an output layer

RCE nets can be cascaded to improve the ability of the network to classify and generalise.

2.4.11 NEOCOGNITRON

The Neocognitron is a neural network that is specifically designed for visual processing. It has features that are based on brain physiology and is hierarchical in nature.

One physiological feature used in this network is the processing of local spatial information, with the neurons only processing a small area on the image and classifying local features. A hierarchical structure can be constructed where the spatial relationship between these features is used in the decision making process. Therefore, this network can consider the structure of the pattern to a limited extent. This hierarchical structure has simple feature-detectors in the low layers that become more complex with level or hierarchy. Similarly, the area considered by the layer projected onto the retina increases with layer, leading to the ability to classify the whole image at a high enough layer.

Another aspect of this local storage of information is to locally classify the features. To accomplish this task two types of neurons are used: simple cells and complex cells. This terminology is loosely based on neurobiological classification of simple, complex, and hypercomplex cells.

A typical structure for a Neocognitron is shown in Figure 2.16.

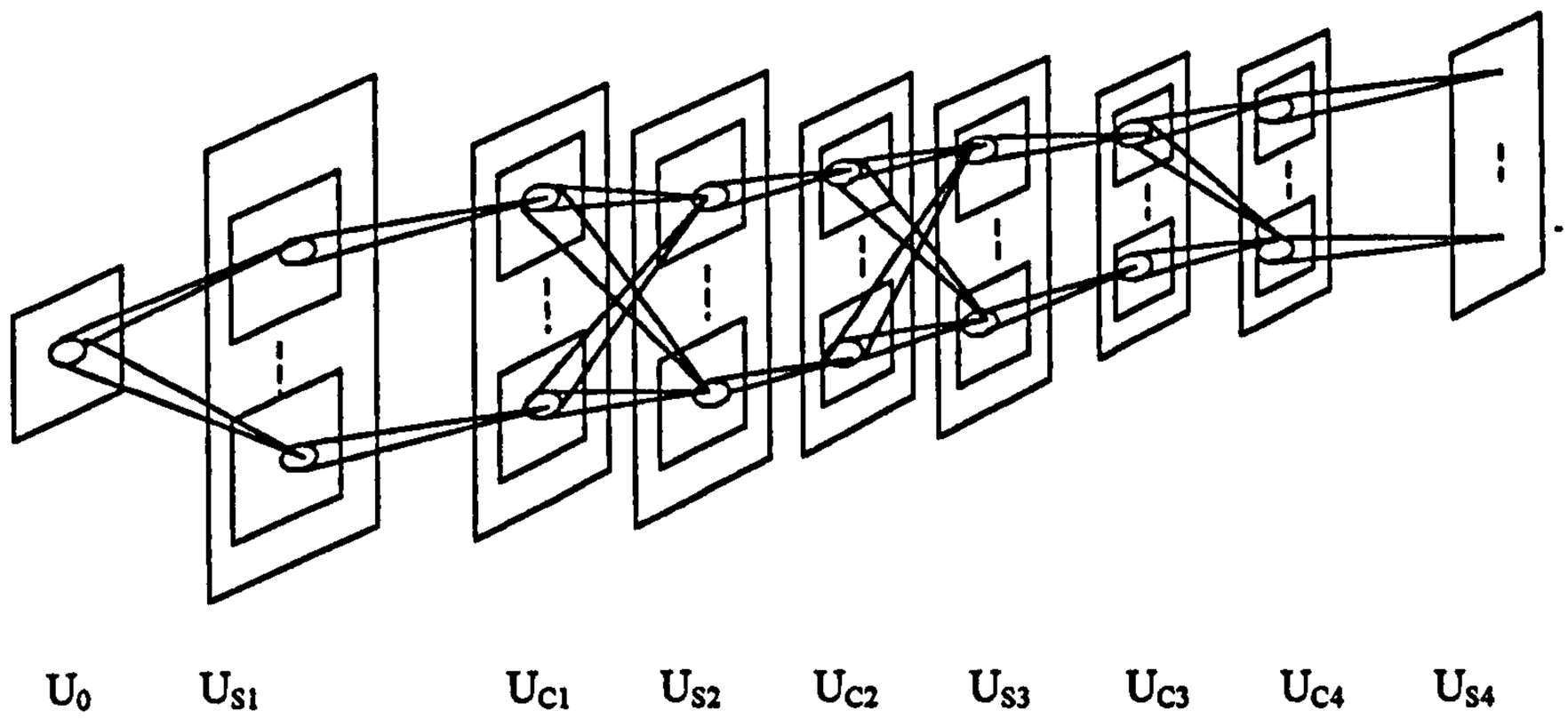


Figure 2.16 Typical structure of a Neocognitron

Each layer in the Neocognitron is constructed from a cascaded connection of modular structures, where each modular structure consists of a layer of simple cells followed by a layer of complex cells. The final complex layer classifies the image. Neurons in a simple layer are known as S cells, and neurons in a complex layer are known as C cells. Each simple or complex layer is made up from sub-groups that are known as planes.

All C cells in a plane have identical weights, and the same synaptic distribution. Each C cell has a receptive field that connects with neurons in the previous layer. This architecture has the property that S cells behave as feature detectors, with each plane detecting a different feature. The receptive field gives the connection a spatial relationship, and the spatial pattern of activity in a plane reflects the spatial distribution of the features in the previous layer.

For the S cells to be able to behave as feature classifiers, they must be able to classify in one layer. This means that the conventional perceptron cannot be used. An inhibitory cell, which is technically part of the complex layer, aids feature classification by providing an

inhibitory signal to S cells. This inhibitory cell has fixed synapses and cannot learn. The S cell is an excitatory neuron and is trained to accept the features, by the modification of the synapses that interact with both C cells and inhibitory cells. This arrangement is shown in Figure 2.17.

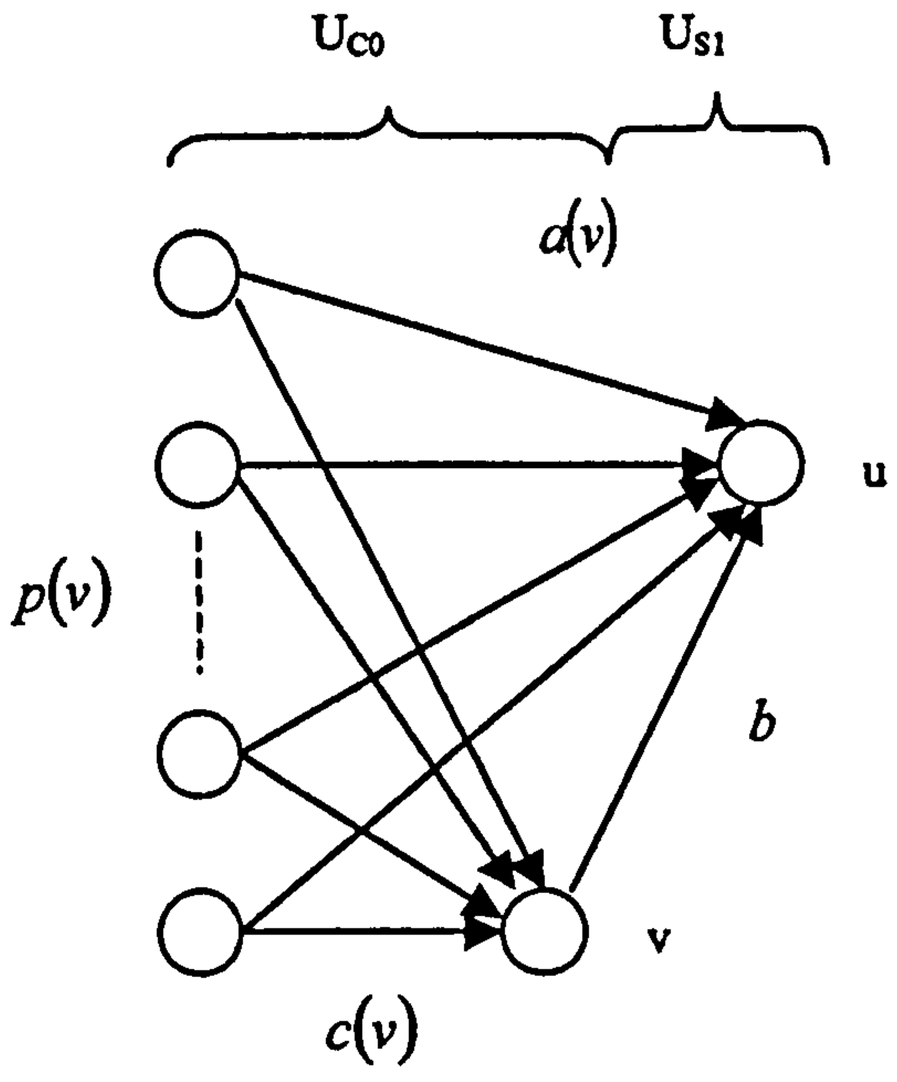


Figure 2.17 Synaptic connections of a simple cell

Fukushima [Fukushima et al. 1983] uses the following notation:

n	- two dimensional co-ordinates of the cell in a plane.
l	- the lth module.
k	- the kth plane.
$U_{Sl}(k,\mathbf{n})$	- the output from the simple cell in module l, plane k, and position **n**.
$u_0(\mathbf{n})$	- output of the photoreceptor at position n.
$u_{Cl}(k,\mathbf{n})$	- the output from the complex cell in module l, plane k, and position n.
v	- an offset vector to n representing the receptive field

$a_l(\kappa, \mathbf{v}, k)$ - the synapse strength of the simple cell in module l from complex plane κ, offset $\mathbf{v}$, and simple plane k.

$b_l(k)$ - the synapse strength of the inhibitory cell in module l in plane k.

The output of an S cell of the kth S plane in the lth module is given by:

$$U_{Sl}(k,\mathbf{n}) = r_l \cdot \phi\left(\frac{1+\sum_{\kappa=1}^{K_{Cl-1}}\sum_{\mathbf{v}\in A_l} a_l(\kappa,\mathbf{v},k)\cdot u_{Cl-1}(\kappa,\mathbf{n}+\mathbf{v})}{1+\frac{r_l}{1+r_l}\cdot b_l(k)\cdot v_{Cl-1}(\mathbf{n})}-1\right) \quad 2.42$$

Where,

r_l controls the intensity of the inhibition with the greater the value the more selective the S cell becomes. Typical values range from 1.0 to 4.0.

$\phi(\cdot)$ is the activation function and is defined as:

$$\phi(x) = x; \quad x \geq 0$$

$$\phi(x) = 0; \quad x < 0$$

v_{Cl-1} is the output from the inhibitory cell, and is given by;

$$v_{Cl-1}(\mathbf{n}) = \sqrt{\sum_{\kappa=1}^{K_{Cl-1}}\sum_{\mathbf{v}\in A_l} c_{l-1}(\mathbf{v})\cdot u^2_{Cl-1}(\kappa,\mathbf{n}+\mathbf{v})} \quad 2.43$$

This gives the output from the inhibitory cell to be proportional to the weighted root mean square of its inputs.

The inhibitory cell weights are determined so that they decrease monotonically with |v|, giving stronger synaptic connection towards the centre. This is similar to some receptive fields in the retina. The values are chosen so that they satisfy:

$$K_{Cl-1} \cdot \sum_{v \in A_l} c_{l-1}(v) = 1 \qquad 2.44$$

The size of the receptive field, A_l, increases with depth.

The C cell layers make up hidden layers in the network and the final layer, though, for the sake of convention, the retina layer is called a complex layer (U_0). Similarly to S cells, C cells have a receptive field so they only consider local information from the preceding layer of S cells. Unlike the S cells, C cells only receive inputs from a small number of planes in the previous S cell layer, for example one or two planes.

The equation for the output of a C cell is given by:

$$u_{Cl}(k,\mathbf{n}) = \psi\left(\sum_{\kappa=1}^{K_{Sl}} j_l(\kappa,k) \sum_{\mathbf{v} \in D_l} d_l(\mathbf{v},k) \cdot u_{Sl}(\kappa,\mathbf{n}+\mathbf{v})\right)_{k=1,2..K_{Cl}} \qquad 2.45$$

Where

$$\psi[x]=\begin{cases} x/(\alpha_l+x), & (x\geq 0) \\ 0, & (x<0) \end{cases} \qquad 2.46$$

α_l is a positive constant with normal values in the range 0.25 to 1.0.

$d_l(\mathbf{v},k)$ is the synaptic weight, it is normal for this value to decrease monotonically with $|\mathbf{v}|$, and apart from layer l = 1, the value is independent of k.

$j_l(\kappa,k)$ denotes the connectivity between planes, where there is connectivity this value is 1, and is 0 for no connectivity.

Menon and Heineman [Menon and Heineman 1988] use a different arrangement for the complex layer. Extra inhibitory cells are provided in the simple layer, the above arrangement for positional tolerance is retained, but the overall C cell equation is similar in structure to the S cell equation.

Two methods of teaching the network are proposed: supervised [Fukushima et al. 1983] and unsupervised [Menon and Heineman 1988].

With supervised learning, a known image is presented and an S cell is chosen in each plane in each layer for teaching. The layers are taught in sequence, starting off at the lowest layer, then the next highest layer until all of the chosen S cells in each layer have been taught. The designated cell in each plane is similarly taught in sequence, so only one cell is

taught at any given instant in time. Initially the weights for the S cells are zero, and the teaching increases the values of both the excitatory and inhibitory weights.

The excitatory weights are taught using the relationship.

$$\Delta a_l\left(\kappa,\mathbf{v},\hat{k}\right)=q_l\cdot c_{l-1}(\mathbf{v})\cdot u_{Cl-1}(\kappa,\hat{\mathbf{n}}+\mathbf{v}) \qquad 2.47$$

The inhibitory weights are taught using the relationship.

$$\Delta b_l\left(\hat{k}\right)=q_l\cdot v_{Cl-1}(\hat{\mathbf{n}}) \qquad 2.48$$

The (positive) term q_l is a gain term and specifies the amount of reinforcement. This method of teaching is similar to Hebbian learning.

For unsupervised learning, a winner takes all strategy is used. This is easier to follow by considering all the S-planes in a layer to be vertically stacked on top of each other. Therefore, a column penetrating the stack of planes will interact with cells having the same spatial position. The system conducts a search process that locates, in each plane, the cell with the maximum output value. By searching all cell positions, maximum values are searched for in each plane. The cell with the largest maximum output is selected in each plane. Provided that the cell has a greater value than all nearby cells in all other planes this neuron is updated, otherwise the maximum neuron is discarded. At the start of the training, the excitatory weights are set to very small randomly set values, and the inhibitory weights are initially set to zero. The same weight-updating equations are used as for supervised learning.

The positional tolerance of the C cells can give a measure of scale, translation, and distortion tolerance.

There have been a number of enhancements to the Neocognitron. One extension by Fukushima [Fukushima 1988, and Fukushima and Imagawa 1993] regards the conventional Neocognitron as a feed forward or afferent system. Another backward or efferent system is used in this extension to provide selective attention. The system can be approximately conceptualised as two Neocognitrons in parallel and working in opposite directions. Extra neurons and connections are added to provide control between the two networks.

One difference between the conventional Neocognitron and the afferent system is that there is lateral inhibition between the S cells, this suppresses the poorly responding neurons and normally only one S cell will respond to an input.

The afferent network and the efferent network both affect each other in the following ways:

- The output from an afferent S cell affects the output from the complementary efferent S cell. This acts as a gate, so a poorly responding afferent S cell will inhibit the output from the efferent S cell. Gating the efferent network in this way elicits a similar response in similar pathways to dominant afferent pathways.

- A no-response detector becomes active if the afferent network has not recognised an image. In this case signals from this detector will lower the threshold of the afferent S cells, allowing the neurons to fire with a greater mismatch between the signal and neuron weights. The threshold signal reaching an S cell depends on the

output from the neighbouring efferent C cells and layer in which it resides. The equations governing the threshold signals are recursive equations that can slowly decay.

- The gain of the afferent C cells is controlled by the output from the corresponding efferent C cells and the search controller. An afferent C cell that has received a strong gain control signal will fatigue unless it continues to receive a large gain signal. This can be used to switch attention to another pattern. A detector monitors the status of the ultimate afferent C layers, and if there is only one cell active in the ultimate layer and the penultimate layer has nearly reached a steady state, a signal is sent to initiate a switch in the attention of the network.

Another extension to the Neocognitron [Wang 1989] uses a self-adaptive network, where the structure of the network evolves to provide a representation that fits the training set. As well as learning being achieved by the change in synaptic weights, learning is also achieved by the growth of neurons. Two growth rules are used: a neuron splitting rule and a neuron formation rule.

Neuron splitting is the production of a neuron adjacent to an existing neuron. Both neurons operate in parallel, facilitating the classification of an extra feature. The conditions for neuron splitting are a true subset of the prominent pathways having large inputs. A prominent pathway is a synapse that has a large weight.

Neuron formation is the process of forming a neuron so that is takes the outputs from a number of inputs, and provides a single output. This neuron operates in series and provides an extra feature that represents the occurrence of the lower level neurons. The rule for

neuron formation is that a neuron is formed if more than one neuron is firing at the same time in a particular layer.

In addition to neuron splitting and formation, the Neocognitron is also modified so that it can classify more than one pattern. This is achieved by the use of an extra layer, called the composite layer, which is connected to the highest complex layer. The neuron formation rule is used to create neurons in the composite layer. If a composite image, containing two or more patterns, is presented then more than one neuron will fire in the last complex layer. This triggers the formation of new neurons that represent this composite image.

The final layer in the Neocognitron has only one neuron for each feature; hence, any information regarding the location of the object has been lost. Using the selective attention model as a basis, a location submodel has been added to give an estimate of the object's location [Kawashima and Ishikawa 1996].

This complex system has been used to perform difficult classification tasks such as classifying joined-up handwriting. The loss of positional information in the final layer is unfortunate and the grafting on of a location submodel can only overcomplicate the model.

Other hierarchical structures have been proposed for example the Cresceptron [Weng et al. 1997].

2.4.12 STRUCTURAL TECHNIQUES

A neural network that receives its input from a structural system has been developed [Ventura and Chen 1996]. The system is built up from a structural system and a number of multilayer perceptron networks in parallel, with each network being designed to handle a

fixed number of image primitives. The structural system identifies: the number of primitives, the length of each primitive, the curvature of each primitive and the angular relationship to the next primitive. Hence, there are three times as many network inputs as there are number of primitives. The network is taught by backpropagation.

The method of classification considers the object boundary in a single stage, and the structure hierarchy of the image is not considered.

2.5 DISCUSSION OF VISION

2.5.1 SUMMARY OF MAMMALIAN VISION

Mammalian vision is the best model of visual processing available. Although it has been investigated for a number of years, it is largely not understood. The system covers a vast range of sophistication, from the behaviour of photoreceptors, which is reasonably understood to the conscious experience of vision, of which virtually nothing has been elucidated. A few key points can be listed which are pertinent to following chapters:

- The behaviour of neurons has been partially elucidated, and it can perform complex processing.

- The structure of the visual system is highly organised.

- Visual processing operations are compartmentalised.

- The organisation of the visual cortex is not completely hierarchical, inasmuch as some layers do not necessarily receive inputs from the previous layer.

2.5.2 DISCUSSION OF NON-NEURAL NETWORK COMPUTER VISION

The methods of template matching, affine transformation matching, Hough transform, Fourier transform, Mellin transform, moments, and structural techniques were previously briefly described.

Template matching is only invariant to translation. To attempt rotation and scale invariance, instead of having one template per object, the templates can be scaled and rotated giving multiple templates for the same object. However this multiplies the number of templates and reduces the performance of the classifier. Apart from the object type, its position (and if implemented, the rotation and scale), no additional information is obtained. The structure of the image is not deduced.

If a template matching system had templates that classified features instead of the whole image, there would be multiple features associated with each image, and knowing the types and location of these features the object could be classified. The classification would be based on the structure of the object so distortion of the object could be tolerated. In addition, the low-level structure of the object would be determined.

By using the relationships between the features the system would be translation and scale invariant. If rotated features were used, the system could also be made to be rotation invariant.

The affine transform matching method can be considered to be a modified template matching classifier, in which an AI search algorithm searches for a match between the object and templates subjected to translation, rotation and scale affine transformations.

When searching for a match between the input pattern and the stored pattern, the criterion used is based on the area of the patterns. The structure of the object is not considered.

This method is not able to determine the structure of the object so, apart from identifying the object, the confidence in the classification and describing the parameters of the affine transformation, no further information about the object can be deduced. Because major modifications to the method would be required to determine the structure of the object, it is considered that a different approach should be pursued.

The Hough transform can be used to identify features in an image. Although by itself it cannot identify anything but the simplest objects, it does provide an example of invariant feature extraction. The amount of processing required depends on the number of degrees of freedom required for the features.

The output from a Hough accumulator has been classified by a neural network. However, the image had to be translated, passed through a $r-\theta$ transform and size normalised before being processed by the Hough transform [Elliman and Banks 1990]. The neural network was used to classify image data from which the position and size information had been removed. Although the Hough transform had identified primitives in the transformed data, the structural information was missing.

The Hough transform could be used to perform feature extraction for use by a higher level processing system that could identify the structure of the object. However, it can suffer from detecting *ghost* features and give ambiguous results. For the scope of this research, a simpler system that does not suffer from these problems would probably be a better approach.

The Fourier transform is a method of converting the image to a translation invariant form, and the Mellin transform converts the image into a scale invariant form. Combining these two methods gives the Fourier-Mellin transform that can be used for translation and scale invariant classification. These techniques give no information about the structure of the object and they are used to classify a complete object. Because these transforms do not naturally lend themselves to identifying the image structure, it was considered that a different approach should be pursued.

Moments can be used for translation, rotation, and scale invariance. However the structural information is lost and the classification process is not able to determine the structure of the object. It was considered that this technique did not have the requirements for further consideration.

Structural techniques try to classify the overall object from its component features. Many systems are based on component rules that were developed by the system builders. One of the objectives of the research was to develop a system that was able to learn in an unsupervised manner. For a system to be relevent to this research, the system must be able to automatically identify the structure of the object.

However, three systems described previously are interesting:

- Feature based [Shepherd et al. 1992]

  This is a good system that is completely separate from neural network techniques. The features are extracted and compared in one stage. When comparing the features, each feature is encoded by calculating a series of metrics denoting the difference between the feature and all the other features. This is repeated for the

other features. Comparisons between all the encoded features of two patterns are made and the minimum classification disparity is used to denote the disparity between the two patterns.

Although the features are extracted from the patterns, they are not combined in a structural or hierarchical manner. This feature-based system is designed to classify complex images and it is not concerned with the structure of the image. This research is interested in the image structure, and the technique is not considered suitable because of the loss in structural information when classifying an image.

- Genetic Programming [Andre 1994]

  This uses a course structural technique and a GP to determine the rules. The object is segmented into four parts and statistics for each segment are calculated. Hence, the GP will produce structural rules based on the statistics of the segments. This causes a loss of information and reduces the effectiveness of any structural decomposition; hence, this method is not suitable.

- Neural network [Ventura and Chen 1996]

  The technique uses pattern recognition techniques to decompose the image into a number of primitives and a multilayer perceptron to classify the image, although the information that reaches the neural network contains a form of structural information. This information is classified as a pattern and the output does not contain any structural information about the image. As the neural network element of this structural technique loses the structural information, it is not suitable for this research.

2.5.3 DISCUSSION OF NEURAL NETWORKS

In the above summary of neural network techniques (Section 2.4), most neurons are based on a very simplistic model for the behaviour of neurons. Most invariant neural network classification methods use invariance pre-processors to provide an output for the classifier. Similarly, most neural networks are taught by supervised learning. This would make the following systems unsuitable for the research: the multilayer perceptron, ART, radial basis functions, high-order neural networks, associative memory and restricted coulomb energy.

The shift invariant layer devised by Elliman and Banks [Elliman and Banks 1990] uses a system that shifts the image. A similar system has also been implemented by Lin [Lin and Wang 1996]. This simple concept is typical of the type of processing used by the retina [Poggio and Christof 1987]. However, rotation and scale invariance is carried out in a non-neural net solution. The shift invariant layer could be suitable for research of this type; however, it might be a better approach to use the inherent parallelism of the neural network to achieve the shift invariance.

The shift, rotation and invariant solution proposed in [Widrow and Winter 1988] is a completely neural network solution, and has been used by Fukumi [Fukumi et al. 1992] and Tanomaru [Tanomaru and Inubushi 1995] for a rotational invariant network. However, the adaptive descrambler removes any structural information about the image. Hence, it is not suitable for this research.

The high-order neural network uses neurons based on a different summing formulation. By careful determination of the weights, the network can be made to classify images in an invariant manner. In addition, this method does not consider any structural detail of the image. Hence, it is not suitable for this research.

Structural techniques have been used [Ventura and Chen 1996] but a conventional neural network was used for image classification. This method was deemed unsuitable for this research because the neural network lost the structural information.

The Neocognitron attempts a hierarchical classification system, where the size and complexity of the features increases with layer. By using local feature classification, the structural relationship between features can be used to classify higher level features. Although the network does not claim to be invariant to shift, scale, or rotation, a degree of tolerance is built into the network. From the point of view of this research, the Neocognitron exhibits the most interesting features.

Further research has shown that the network is not intrinsically shift invariant [Barnard and Casasent 1990], and Menon [Menon and Heinemann 1988] estimates that a Neocognitron with at least 15 levels would be required to provide shift invariance for a 128 × 128 image.

The manner in which the hierarchical structure is implemented means that in the final layer, there is only one neuron for each object. Therefore, any positional information is lost. Attempts have been made to recover this information [Kawashima and Ishikawa 1996], but it is better not to lose it in the first place.

It is considered that the Neocognitron has many useful features for example, it has localised receptive fields, and weight sharing. These and other Neocognitron techniques could be used in this research.

Many synapse updating formulae were based on Hebbian learning, although one notable exception was Kohonen learning. This had the advantages that the value for a weight could

decrease as well as increase, and the value for the weight asymptotically approaches the value of the input. The final weight value can easily be related to the neuron's physical environment.

3.0 SHIFT INVARIANT NEURAL NETWORK

3.1 CHAPTER CONTENTS

This chapter introduces the elements of a network that can perform both invariant and structural classification. These ideas are developed to produce a neural network system capable of shift invariant classification. This work has been previously published [Grimes et al. 1996].

3.2 CHARACTERISTICS OF THE SYSTEM

The proposed system will have elements of both structural pattern recognition and neural networks. To achieve these requirements considering the previous discussions, the following characteristics for the system are listed below:

- The network should be hierarchical

  One aim of the research is to produce a network that will perform structural decomposition of the image. The lowest layer will recognise simple features, and the highest layer will classify the whole image. The output from the intermediate layers should be recognisable features, with the higher layers classifying increasingly complicated features. This hierarchical property is characteristic of the Neocognitron. It is legitimate to receive input from any lower level layer, not just from the previous layer.

- Local information must be used

  Neurons will not be fully connected as in the case of the multi-layer perceptron. Their inputs will come from a receptive field. This will allow the network to retain

the relationship between features, and will facilitate a hierarchical structure, as is the case for the Neocognitron. The network should be arranged so that the higher the layer, the larger the area on the retina *seen* by the neurons.

- Neurons should be capable of feature extraction

  With conventional perceptrons, one neuron by itself does not have enough discrimination capability to be able to classify certain features by itself. For this reason neurons are arranged in layers to provide better classification for more complicated problems, for example the exclusive-OR problem. For the network to perform structural decomposition, it would be advantageous for a neuron to be able to classify a salient feature in the image. Perceptron type neurons would be unsuitable for this task, so a neuron with better classification properties would have to be used.

- Unsupervised learning should be used

  In trying to produce a network that will automatically deduce the structure of the image, it would be desirable for the network to learn by unsupervised learning. This means that the learning process involves automatically identifying salient features in the image and assigning neurons to these features. Developing algorithms to accomplish this task was problematic.

3.3 OVERVIEW OF THE NETWORK

Template matching was discussed in Chapter 2. Templates, which are much smaller than the retina, and having value patterns describing features, mathematically scan the retina image. The output from the template is a measure of how *close* the template is to a feature in the image. The greater the output, the closer are the template and feature patterns. By

scanning the image with multiple templates, and noting the position and output value when the output is a maximum, the following information can be determined:

- The feature in the image

  This would be given by the feature with the greatest output.

- The position of this feature

  This would be given by the position of the feature that classifies the image.

- The confidence of the classification

  This is given by the output value of the determined feature at the feature position.

This method has some useful characteristics:

- The classification is shift invariant

- The location of the feature can be determined during the classification

These features can be useful for the first stage of a neural network hierarchy that can classify objects in a structural manner.

Chapter 2 gave simple descriptions for the behaviour of biological neurons, and also contained descriptions for a number of mathematical models for neurons used in pattern classification. By comparing the two descriptions, it appears that the mathematical models are gross simplifications of the processing carried out in biological neurons. One reason

for this simplification is the current lack of understanding of the behaviour of biological neurons.

Artificial neural networks seem to have characteristics of simple neurons combined with complicated mathematically derived learning mechanisms, where the location of this learning mechanism is separate from the neurons. Indeed most diagrams of artificial neural networks do not show the learning mechanism. The biological basis of this learning is just becoming known, for example the discovery of second-messenger systems [Kandel and Hawkins 1992], and the local-interaction model of learning proposed by Alkon [Alkon 1989].

Although there are many *standard models* for artificial neural networks, these models seem to have little in common with the biological neurons that they try to emulate. This has been summarised in [Suárez Arajo 1997]:

No models to date have been able to duplicate the powers and profound capacities of the human brain. Neuronal models do not explain how the brain generates our cognitive and perceptual experience.

It is legitimate to develop a new artificial neural network with characteristics suitable for the current problem.

For the above reasons, it was decided to construct artificial neurons with similar processing characteristics to templates.

By considering one spatially fixed template as a neuron, the neuron can be considered to have the following characteristics:

- It uses local information

  The neuron receives its input from a limited region called a receptive field, and the output represents the closeness of the input pattern to the internal state of the neuron, with values ranging from 0.0 to 1.0. The greater the output the better the classification.

- The neuron is capable of feature extraction.

  Using numerical values ranging from 0.0 to 1.0 throughout the system combined with the feature extracting property, a hierarchical system can be constructed.

Having just one neuron at a fixed location will only give the confidence that a particular feature is at that location. Having more than one neuron operating in parallel, with each neuron tuned to detect a specific feature, enables more than one feature to be classified. This arrangement is known as a feature detector, and is shown in Figure 3.1.

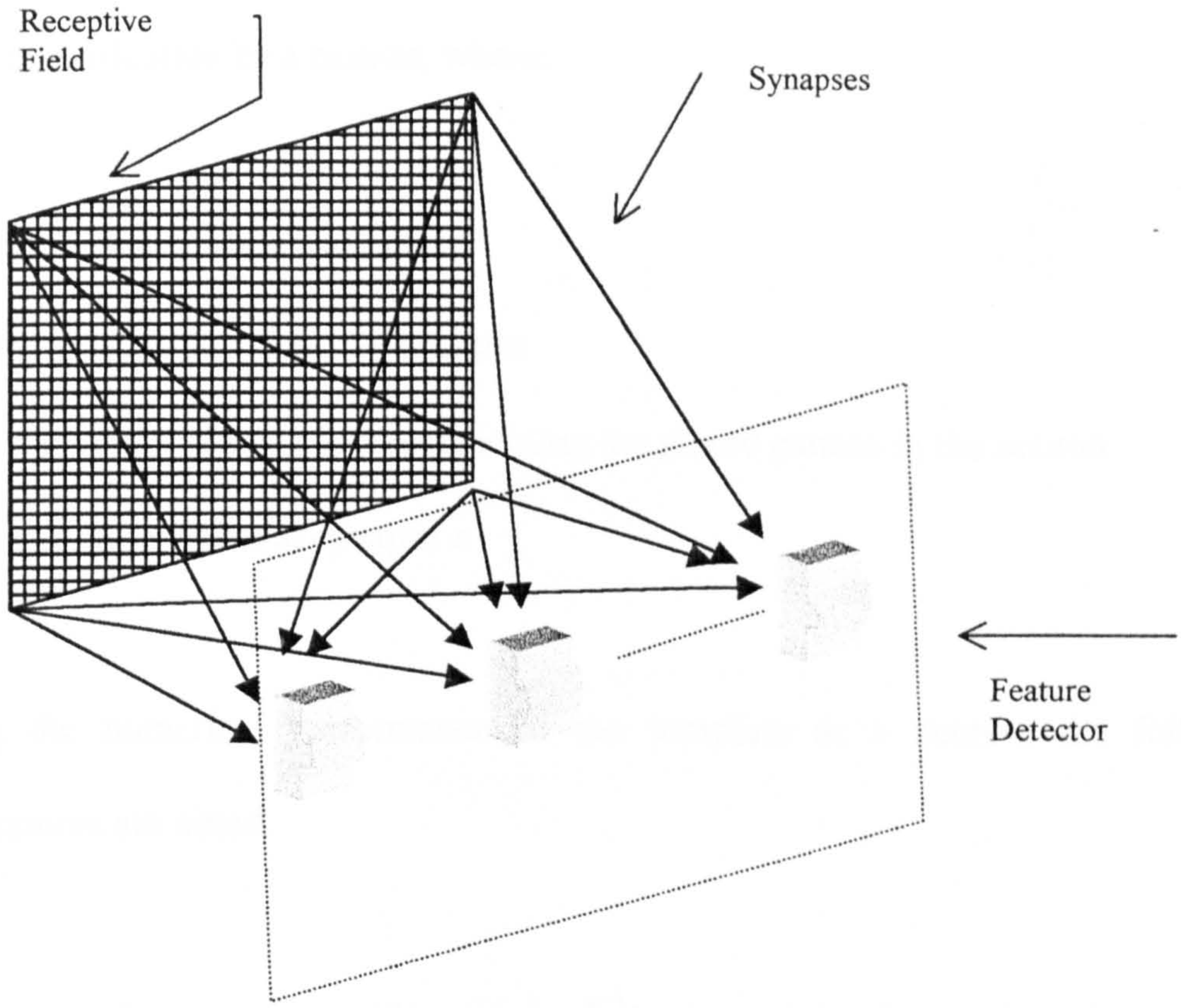


Figure 3.1 Feature detector

To provide feature detection across the whole of the retina, the feature detector is replicated across the retina with each detector offset from its neighbour by one pixel in the horizontal, vertical, or both directions. This is equivalent to the scanning of the retina by the templates.

3.4 DESCRIPTION OF THE NETWORK

The error between a template and the pattern is given by equation 2.12, and is repeated here for clarity.

$$d^2(\mathbf{y}) = \sum_{\mathbf{x}} (f(\mathbf{x}+\mathbf{y}) - t(\mathbf{x}))^2 \qquad 3.1$$

Although equation 3.1 represents classification by template matching, it can also be used to represent classification by a neuron, where:

y is the location of the neuron

x is the relative location of a synapse

$t(\mathbf{x})$ is a weight at synapse **x** representing the stored pattern in the neuron

$f(\mathbf{x})$ is an input value at synapse **x**

Considering the numerical performance of the template as a neuron, the following unhelpful features are noted:

- The neuron's output range ($0 \le d^2(\mathbf{y}) \le \sum_{\mathbf{x}} 1$) depends on the number of synapses.
- The higher the output value the greater the mismatch between the two patterns, which is the opposite to the operation of most neurons.

The template matching equation can be converted into a form suitable for acting as a neuron by normalising the output and subtracting from 1, giving Equation 3.2 in vector form and Equation 3.3 in scalar form;

$$Net = 1 - \frac{|\mathbf{x} - \mathbf{w}|}{|\mathbf{I}|} \qquad 3.2$$

$$Net = 1 - \frac{\sqrt{\sum_{j=1}^{n}(x_j - w_j)^2}}{\sqrt{\sum_{j=1}^{n} 1}} \qquad 3.3$$

An alternative way of viewing the behaviour of the neuron is to consider a hypercube in which each side has a length of 1.0. The number of dimensions making up the hypercube is the same as the number of synapses. Both **x** and **w** are vectors describing the co-ordinates of points within the hypercube. The distance between the two co-ordinates (or Euclidean Norm) is given by the numerator of Equation 3.2, and is the measure of the error between the two applied and stored patterns.

A diagonal line across opposite corners of the hypercube represents the maximum dimension within the hypercube, and is given by the denominator of equation 3.2.

Equation 3.2 can be considered as the current distance error between the two patterns divided by the maximum possible pattern, subtracted from 1.0.

When the two co-ordinates are coincident the error is zero, giving a value for Net of 1.0. However, Net will only become zero when the two co-ordinates are at opposite corners. As this is deemed to be an unlikely situation, the value for net will rarely approach zero. This means that the pattern rejection performance will not be very good. To improve on the ability of a neuron to reject non-matching features, extra compensation terms were added. The weight associated with synapse j, w_j, will have a range between 0.0 and 1.0, and the maximum possible error will occur when the input value has an extreme value. This gives the maximum error (expressed as a distance length) for this synapse of

$$l_j = \max(1 - w_j, w_j - 0) \qquad 3.4$$

For a one-dimensional hypercube, the maximum possible error length for the neuron is given by equation 3.4. Higher dimensional hypercubes will have an increased volume and similarly the corresponding maximum diagonal is increased. Putting this compensation into equation 3.3 gives;

$$Net = 1 - \frac{\sqrt{\sum_{j=1}^{n}(x_j - w_j)^2}}{\sqrt{\sum_{j=1}^{n}(l_j)^2}} \qquad 3.5$$

Because the neurons use local information, a multi-layered network made from such neurons is expected to be *massively parallel*, and neurons are expected to provide good classification when having a large number of synapses.

These neurons will produce analogue level outputs and it is expected that most neurons will not totally accept or totally reject a pattern. They will partially classify giving an output about mid-point in the signal range, say 0.5. A large number of partially classifying neurons and only a small number of totally accepted or rejected neurons will mean that the neurons in a following layer could find classification difficult. This is because the effect of the few classified signals is being swamped by the many partially classified signals. Figure 3.2 shows the possible inputs to a higher layer neuron when two different patterns are presented. The abscissa shows synapse number and the ordinate shows the input value. It can be seen that only a few salient inputs are required to classify the two patterns, the other inputs providing very little information to the neurons.

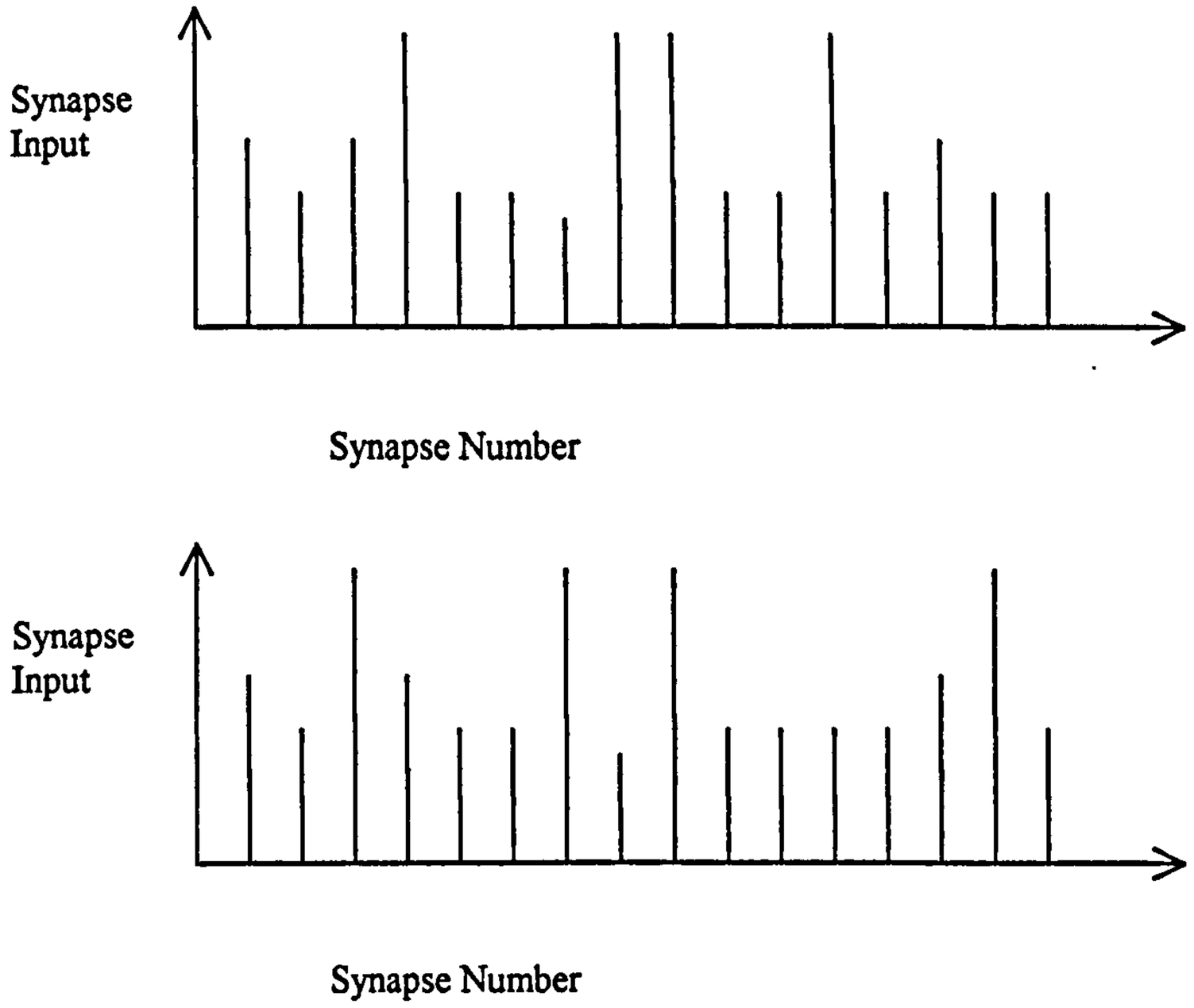


Figure 3.2 Possible inputs to a neuron

The effect of swamping a few classified inputs by a large number of partially classified inputs can reduce the performance of the network. This effect can be reduced by letting the synapses which provide little information to the system have little effect on the classification performance of the neuron.

Another way of stating this is to say that synapses that receive little novelty will atrophy. In the extreme case, a synapse that receives a constant signal should atrophy to such an extent that it will have no effect on the performance of the neuron; in other words, it has died.

This effect can be modelled by having an extra term to equation 3.5, which is called the conductance, see equation 3.6.

$$Net = 1 - \frac{\sqrt{\sum_{j=1}^{n} (x_j - w_j)^2 c_j}}{\sqrt{\sum_{j=1}^{n} (I_j)^2 c_j}} \qquad 3.6$$

Where the conductance is given by;

$$0 \leq c_j \leq 1$$

The conductance term, c_j, is present in both the denominator and the numerator. In the numerator, it will have the effect of reducing the overall sum making *Net* larger, whereas in the denominator it will have the effect of reducing the volume of the hypercube and decreasing *Net*.

A simple analysis of the above neuron can demonstrate the effect of the conductance. Assume that the neuron receives inputs that are restricted to 3 values: 0.0, 0.5, and 1.0, where the synapses either receive inputs which switch between 0.0 and 1.0, or are maintained at 0.5. For a neuron with n inputs, from which u inputs are varying and v inputs are fixed, Equation 3.6 can be rewritten by separating the synapses into varying and fixed input groups, giving Equation 3.7.

$$Net = 1 - \frac{\sqrt{\sum_{j \in u} (x_j - w_j)^2 c_j^+ + \sum_{j \in v} (x_j - w_j)^2 c_j^-}}{\sqrt{\sum_{j \in u} (l_j)^2 c_j^+ + \sum_{j \in v} (l_j)^2 c_j^-}} \qquad 3.7$$

Where,

c^+ is the conductance for varying inputs

c^- is the conductance for fixed inputs

After a period of training, the values for the weights will approach that of the inputs. This affects the fixed inputs term of equation 3.7, giving

$$\sum_{j \in v} (x_j - w_j)^2 c_j^- \to 0 \qquad 3.8$$

Equation 3.7 reduces to Equation 3.9

$$Net = 1 - \frac{\sqrt{\sum_{j \in u} (x_j - w_j)^2 c_j^+}}{\sqrt{\sum_{j \in u} (l_j)^2 c_j^+ + \sum_{j \in v} (l_j)^2 c_j^-}} \qquad 3.9$$

The effect of the conductance can be shown by considering two sets of values for c^+ and c^-. A learning rule can give conductance compensation, that is a conductance of 1.0 for the changing inputs and 0.0 for the fixed inputs. Without conductance compensation, both c^+ and c^- have values of 1.0, and the value for *Net* is given by *Net'*. With conductance compensation c^+ is 1.0 and c^- is 0.0, giving a value for *Net* of *Net''*. Hence without compensation

$$Net' = 1 - \frac{\sqrt{\sum_{j \in u} (x_j - w_j)^2}}{\sqrt{\sum_{j \in u} (l_j)^2 + \sum_{j \in v} (l_j)^2}} \qquad 3.10$$

With compensation

$$Net'' = 1 - \frac{\sqrt{\sum_{j \in u} (x_j - w_j)^2}}{\sqrt{\sum_{j \in u} (l_j)^2}} \qquad 3.11$$

By comparing the two values for *Net*

$$Net' \geq Net'' \tag{3.12}$$

For close matching of the input and weight vectors the values for the numerators will be very small, and conductance compensation will have negligible effect. This effect reduces with the closeness of the two vectors until at perfect matching they both give the same value of 1.0.

With a large difference between the two vectors the difference between the values for Net increases. The numerator gives the error term for the two vectors. As these are varying inputs with values of 0.0 and 1.0, the maximum value for this is geometrically given by a line linking corners of the hypercube. For increasingly large mismatches, the numerator is given by

$$\sqrt{\sum_{j \in u}(x_j - w_j)^2} \rightarrow \sqrt{\sum_{j \in u}(l_j)^2} \rightarrow \sqrt{u} \tag{3.13}$$

Using this limiting value in Equations 3.10 and 3.11 gives

$$Net'|_{large\ error} = 1 - \frac{1}{\sqrt{1 + \frac{\sum_{j \in v}(l_j)^2}{u}}} \tag{3.14}$$

$$Net''|_{large\ error} = 0 \tag{3.15}$$

Therefore, the conductance compensation improves the rejection of misclassified patterns.

It was stated in Section 3.6 and by Wasserman [Wasserman 1989] that a non-linear activation function was required for perceptron type neurons when they were to be cascaded into more than one layer. Some learning algorithms necessitate the need for the activation function to be differentiable, for example back-propagation. One suitable form of activation function that meets both requirements is the sigmoidal function used in back-propagation. A neuron that does not need to be constrained by the above conditions can have a wider choice for the activation function.

Perceptron type neurons have a linear summing unit followed by a non-linear activation function, whereas inspection of Equation 3.6 reveals a non-linear summing unit. Ignoring the effect of the learning algorithm, the neuron as specified by Equation 3.8 could be cascaded into layers without an activation function.

An activation function could benefit the performance of the neuron by increasing the contrast between well and poorly classified features. Another benefit could be the introduction of a tolerance; that is, the neuron will register maximum output if there is a slight mismatch between the input pattern and the weights.

The next section will introduce the learning algorithms, which do not require a differentiable activation function. From a wide choice of possible activation functions, experiments were carried out with an activation function made up of two components.

For $0 \leq Net < \tau$

$$Output = 1 - \sqrt{\frac{\tau - Net}{\tau}} \qquad 3.16$$

For $\tau \leq Net \leq 1$

$$Output = 1 \qquad 3.17$$

The activation function is shown in Figure 3.3.

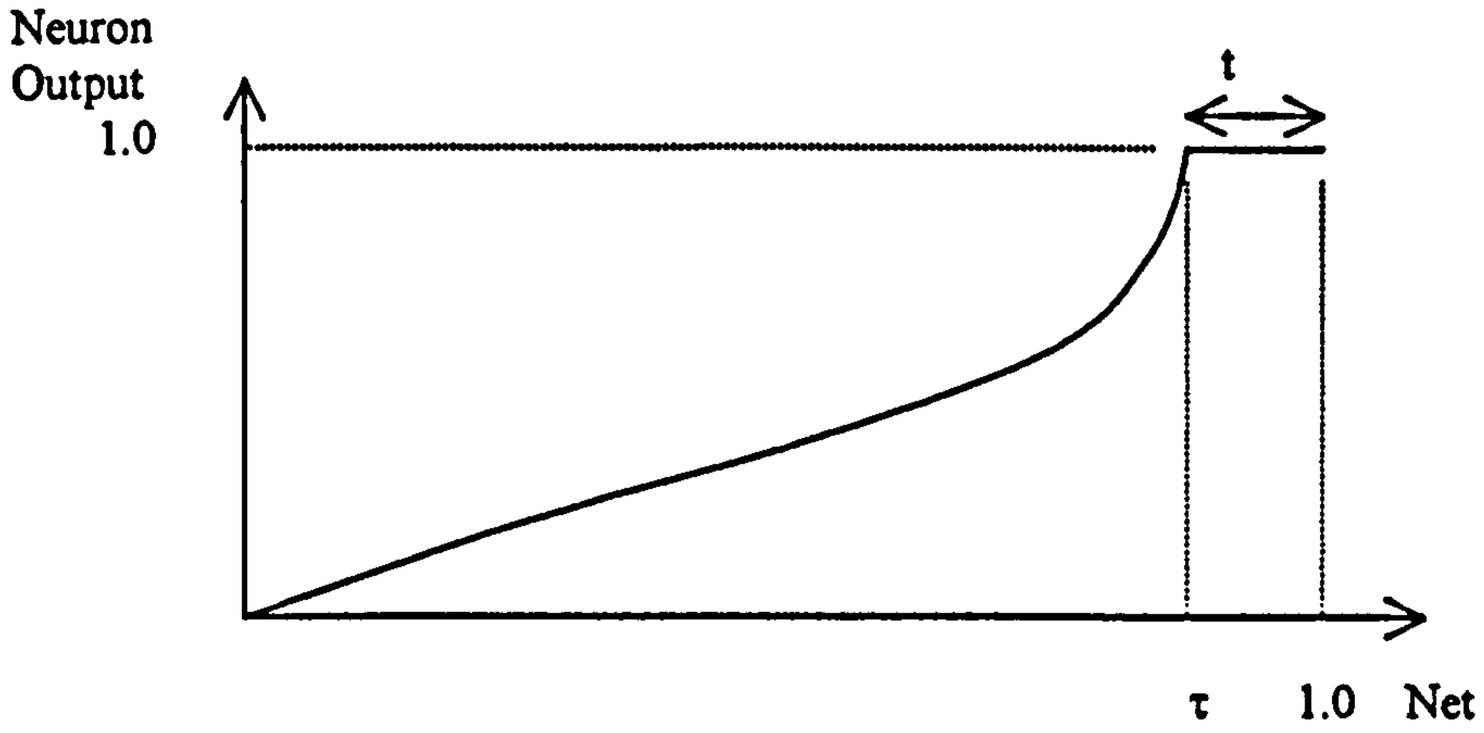


Figure 3.3 Activation function

The neurons are connected together to form a network in a structured manner. A co-ordinate system is used to define the position of various items: units of rows and columns are used to denote distances in two directions. This is shown in Figure 3.4

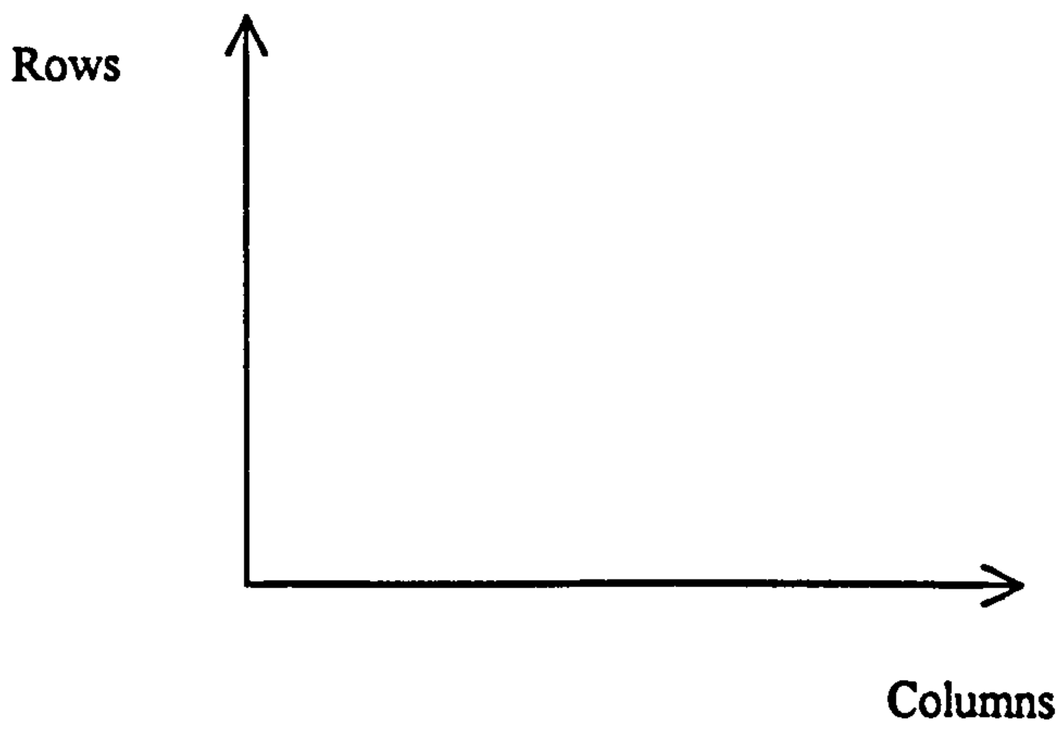


Figure 3.4 Co-ordinate system used

When describing the connectivity of the layers of the neurons to make a neural network, the layer number is given by the letter k. The first layer has a value of 1, second has a value of 2, etc. The input layer is denoted by a value for k of 0.

Layers of neurons are arranged in rectangular grids and the size is denoted by the number of rows and columns making up the rectangle. For layer k

- M_k is the number of columns of neurons in a layer
- N_k is the number of rows of neurons in a layer

The neurons have a rectangular receptive field, the size of which is constant for all neurons in the layer. In this type of network, the receptive field does not indicate the number of synapse rows and columns but the number of feature detectors that neurons process. For layer k

- m_k is the number of receptive field columns
- n_k is the number of receptive field rows

As stated in Section 3.3 the neurons which have the same receptive field act together to form a feature detector. The number of possible features that can be detected in a layer is constant. For layer k

- f_k is the number of features

The retina is deemed to have only one feature, the pixel value, giving $f_0 = 1$.

The total number of neurons in layer k is given by Equation 3.18

$$T_k = M_k N_k f_k \qquad 3.18$$

Section 3.3 describes how the feature detectors give complete spatial coverage by having the feature detectors offset from each other by one pixel. This constraint will determine the number of rows and columns of neurons in all layers in the network, as given by Equations 3.19 and 3.20.

$$M_{k+1} = M_k - m_k + 1 \qquad 3.19$$

$$N_{k+1} = N_k - n_k + 1 \qquad 3.20$$

The neurons in a feature detector are arranged in a horizontal row. A neuron in a following layer will require the following numbers of synapse rows and columns; the number of synapse columns is given by

$$m'_{k+1} = m_{k+1} f_k \qquad 3.21$$

The number of synapse rows is given by

$$n'_{k+1} = n_{k+1} \qquad 3.22$$

Figure 3.5 shows the connections between layers in a network.

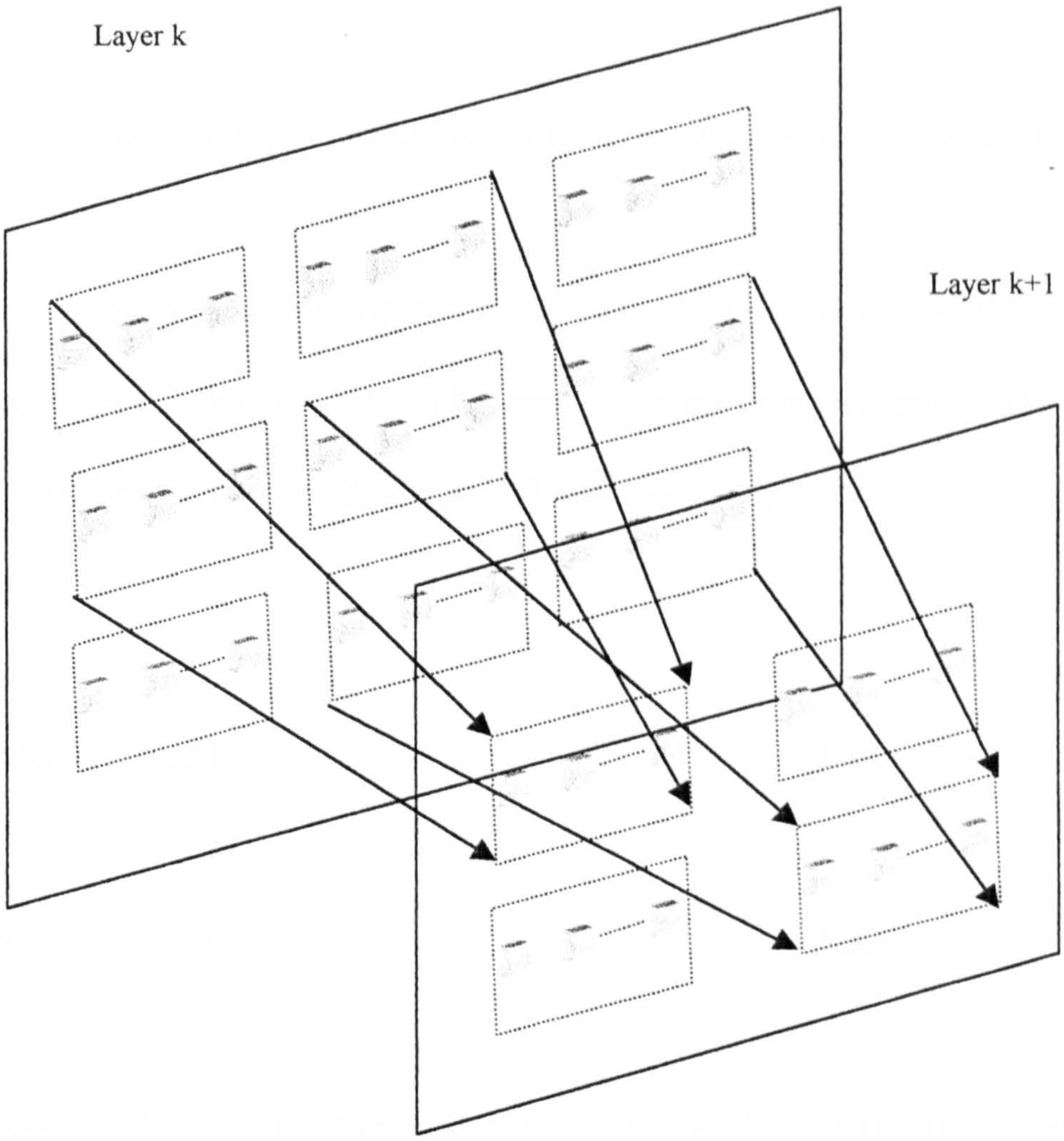


Figure 3.5 Network connections

A point on the retina can be transformed to a point in the first layer then to the second layer, third, etc, until the final layer has been reached. This transformation maintains the same position of interest independent of the layers, but because the number of neurons varies with layers the neuron's co-ordinates will change with layer.

The number of feature detector rows and columns decreases with layer. Because this decrease is symmetrical about the centre of the retina, the following equations relate the transformed co-ordinates from one layer to the next.

$$x_{k+1} = x_k + 0.5(M_{k+1} - M_k) \qquad 3.23$$

$$y_{k+1} = y_k + 0.5(N_{k+1} - N_k) \qquad 3.24$$

The x co-ordinates are expressed in columns and the y co-ordinates are expressed in rows.

Because of the hierarchical nature of the network, as the layer increases, the effective receptive field on the retina increases. The size of the effective receptive field can be projected forward. Rearranging Equations 3.23 and 3.24 to project towards the retina gives;

$$x_k = x_{k+1} + 0.5(M_k - M_{k+1}) \qquad 3.25$$

$$y_k = y_{k+1} + 0.5(N_k - N_{k+1}) \qquad 3.26$$

These equations transform the position of the neuron; this position is also the centre of the receptive field. For a neuron in layer $k+1$, the receptive field can be transformed to layer k by:

$$x_{k\,\min} = round\left(x_{k+1} + 0.5\left(M_k - M_{k+1} - \left(\frac{m_{k+1}}{f_k} \right) + 1 \right) \right) \qquad 3.27$$

$$x_{k\,\max} = round\left(x_{k+1} + 0.5\left(M_k - M_{k+1} + \left(\frac{m_{k+1}}{f_k} \right) - 1 \right) \right) \qquad 3.28$$

$$y_{k\,\min} = round\left(y_{k+1} + 0.5(N_k - N_{k+1} - n_{k+1} + 1) \right) \qquad 3.29$$

$$y_{k\,\max} = round\left(y_{k+1} + 0.5(N_k - N_{k+1} + n_{k+1} - 1) \right) \qquad 3.30$$

In the more general case of projecting a receptive field forward, Equations 3.27 to 3.30 can be modified to project the extremities of the receptive field, giving;

$$x_{k\min} = round\left(x_{k+1\min} + 0.5\left(M_k - M_{k+1} - \left(\frac{m_{k+1}}{f_k} \right) + 1 \right) \right) \qquad 3.31$$

$$x_{k\max} = round\left(x_{k+1\max} + 0.5\left(M_k - M_{k+1} + \left(\frac{m_{k+1}}{f_k} \right) - 1 \right) \right) \qquad 3.32$$

$$y_{k\min} = round\left(y_{k+1\min} + 0.5\left(N_k - N_{k+1} - n_{k+1} + 1 \right) \right) \qquad 3.33$$

$$y_{k\max} = round\left(y_{k+1\max} + 0.5\left(N_k - N_{k+1} + n_{k+1} - 1 \right) \right) \qquad 3.34$$

This network has some features that are similar to the Neocognitron [Fukushima et al. 1983, Fukushima 1988, and Fukushima and Imagawa 1993], for example:

1. Both networks are hierarchical in structure

2. Features are recognised as entities in their own right in each layer, as opposed to patterns in other networks such as the multi-layer perceptron.

As a corollary of 2, the neurons in each layer must be capable of classification in their own right. The Neocognitron uses simple and inhibitory cells working in opposition, whereas the feature extracting neurons presented here use an error metric.

With the Neocognitron, the features are stored in different planes and the following neurons have a receptive field that reads a spatially identical area in each plane. The arrangement described here has all the feature neurons for a certain location arranged adjacent to each other in groups called *feature detectors*.

3.5 LEARNING ALGORITHMS

The form of the neuron equation has two parameters, the synapse weight and conductance, which determine which feature the neuron will detect. Both parameters must be taught in a manner allowing the neuron to adjust to its environment. A number of different learning algorithms are available and are normally based around the characteristics of the neuron.

The neuron used in this chapter has the following weight and conductance characteristics:

- The synapse weight should approach the value of a representative average of the input values of the feature class members.

- The synapse conductance should approach an average value of the range of input values of the inputs.

Both of the above conditions imply that the values for these parameters can decrease as well as increase. This will rule out Hebbian learning [Kohonen 1988].

It was previously mentioned that the network should be capable of unsupervised learning, this will rule out the algorithms used for back-propagation.

One form of learning that fits the above criteria is Kohonen learning. Applying this to the weights gives Equation 3.35

$$w_{j_{k+1}} = w_{j_k} + K_s\left(x_j - w_{j_k}\right) \qquad 3.35$$

The value K_s represents the rate of learning where:

For $K_s = 0$

$$w_{j_{k+1}} = w_{j_k} \qquad 3.36$$

There is no learning.

For $K_s = 1.0$

$$w_{j_{k+1}} = x_j \qquad 3.37$$

The synapse weights will instantly approach the input feature.

Both of the above learning rate values are useless for training purposes, the condition K_s = 0 giving no learning, and K_s = 1.0 setting the neuron weights to the value for the last presented value in the class. This is fine where all values in the class are the same, but for variations in the class, the values will not approach a representative average. Learning rate values of about 0.5 were used in the experiments.

When considering updating the conductance, a measure for the variety of input into a synapse must be defined. Because the synapse weights are adjusted towards the target feature values, they can be considered to have a memory of the values arriving at a synapse. Interrogating the synapse weights can give a measure of the variety of the input

signals. One measure of the variety of inputs can be the range of weight values, giving Equation 3.38

$$R_j = \max\left(w_{jf}\Big|_{\forall f}\right) - \min\left(w_{jf}\Big|_{\forall f}\right) \qquad 3.38$$

The conductance is defined as the square of the range. As for the synapse weights, Kohonen learning is used to train the conductance, giving Equation 3.39

$$c_{jk+1} = c_{jk} - K_c\left(c_{jk} - R_j^2\right) \qquad 3.39$$

The term K_c has the range $0.0 \le K_c \le 1.0$, and it is called the atrophy because it atrophies the synapses that do not have much input variation.

Once a neuron has been updated by any of the above rules, the values are copied to all of the same feature neurons at different locations. This is known as weight sharing. When implementing this network in a computer simulation this property can be extremely useful. The output values for each neuron must be stored, whereas all the synapse data only need to be stored once for each feature. This can result in a massive saving in the data requirements, because the synapse data requires much more storage than the neuron's output data.

Before the network can be taught, the synapse values must be initialised to the following values:

- Weights are randomly chosen to lie in the range 0.0 to 1.0.
- Conductances are set to 1.0.

The initial value for the weights is at variance with the initialisation condition for a number of different networks, for example back-propagation in which the weights are set to small values. Setting the weights for this neuron to small values would bias the neuron to detect features with low signal values. This is because this neuron classifies both inputs of 0.0 and 1.0 in an equivalent manner. An unlit part of the background is deemed to be as valid as a completely lit large portion of the object. Initialising all the features in a random manner between 0.0 and 1.0 will give each feature an equivalent chance of being selected by a neuron.

Unsupervised learning was used to teach the network, and a competitive learning strategy was employed to determine which neurons to update. This is briefly shown for the presentation of one image:

Present image

FOR first to last layers **DO**

BEGIN

Search network for neuron with largest output

Update this neuron

Copy synapse values to other *same feature* neurons

END

This would be repeated for many presentations of many images, giving two extra program loops.

For a number of reasons, the above teaching rules combined with the simple teaching algorithm did not produce very good results. One important reason was the combined effect of translation invariance and unsupervised learning, in which the neurons learn features in the image that give a good output. It has been observed that the neurons can learn a feature in one image that is also present in a different image at a different location. The neurons that learn these features will dominate the learning process, leading to the situation where a few features are learnt and other features are missed.

The classification of the different features by the same class of neuron is shown in Figure 3.6.

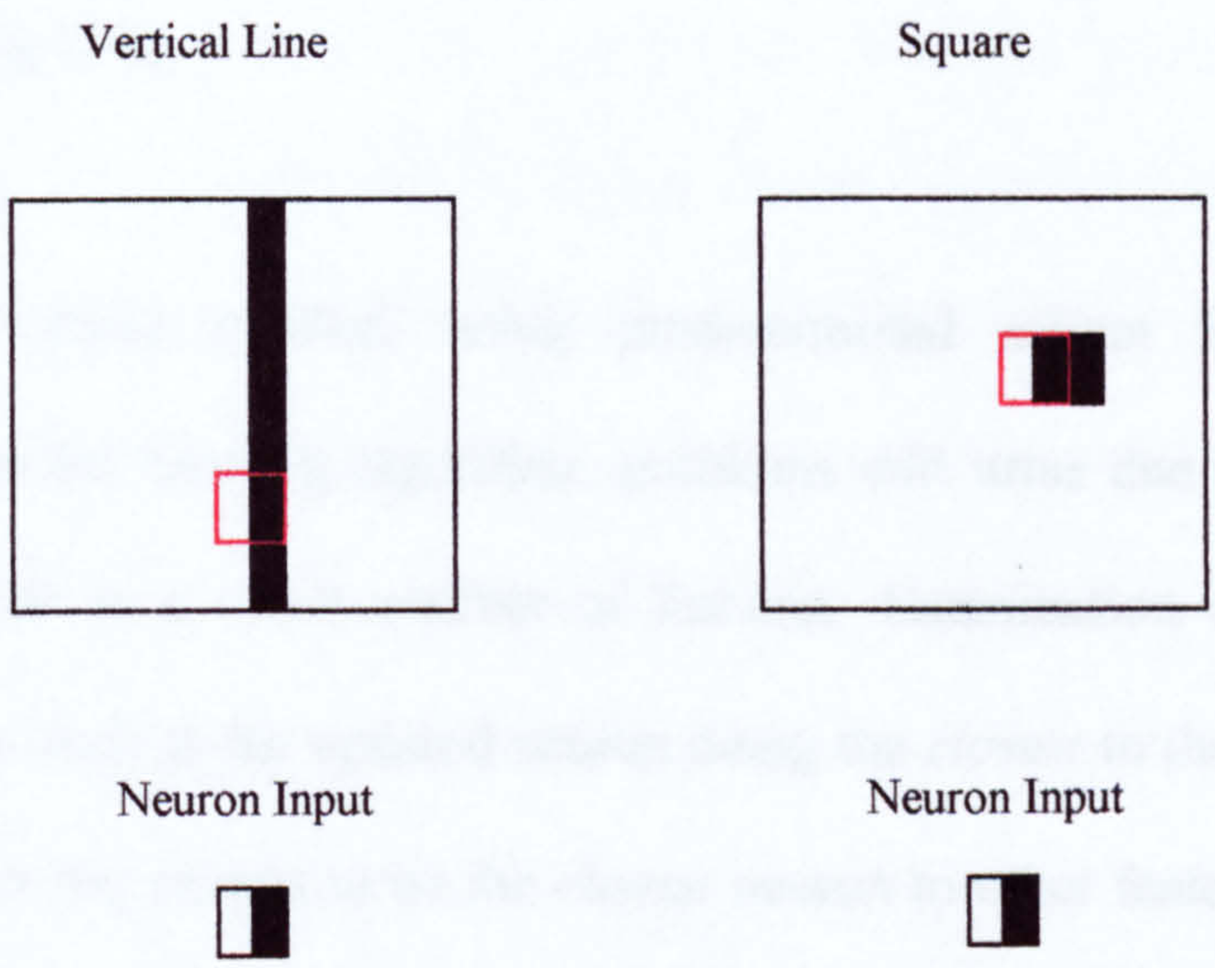


Figure 3.6 Different images exciting the same feature neurons

One way of overcoming this problem is to give the location of a salient feature in the image along with the image. This is known as the image hot spot, and is given in retina co-ordinates. Using Equations 3.23 and 3.24 the co-ordinates on the hot spot are transformed from retina co-ordinates to the co-ordinates of higher layers until the maximum layer is reached.

The receptive field is calculated for this layer, and using Equations 3.31 to 3.34 the region of neurons in lower layers that affect this neuron is determined. The region of neurons in a layer containing neurons that affects the hot spot neuron in the maximum layer is known as the capture area. As the network layer number reduces, the size of the capture area increases.

When determining which neuron to update by the competitive learning strategy, instead of searching the whole layer only the neurons in the capture area are examined. In the highest layer, only one feature detector is examined, and the search algorithm will only determine which feature must be taught. The much reduced search area in each layer leads to a much reduced teaching time.

Although the above method, using predetermined salient features, improves the performance of the learning algorithm, problems still arise due to the learning process being dominated by a small number of features. Examination of the learning process revealed that as well as the updated neuron being the *closest* to the presented feature, it is also possible for the neuron to be the *closest* neuron to other features. If any one of these other features is presented, the same neuron will also try to learn these features, thus preventing other neurons from learning them.

The following discussions will mention synapse space; this is the space represented by the synapses with each synapse adding an extra dimension. Strictly speaking, the analysis using synapse space is only valid for the uncompensated type of neuron as given by Equation 3.2. Both error and conductance compensation will affect the analysis, but the overriding principles are still primarily governed by the error between the feature input and synapse weight vectors. Any following mathematics will also reflect this simplification.

The output from a neuron depends on the geometrical arrangement of the input feature and the synapses, with the closer the two co-ordinates the larger the output.

To illustrate the effect of dominance by a few neurons, the following illustration assumes that there are two neurons in each feature detector, each neuron having two synapses, and two possible input features. Ideally, each neuron would learn a different feature. Figure 3.7 shows the synapse space for the neurons. The synapse space can be fully represented by a square because the neurons have two synapses.

The position invariance property of the network means that positions of all features presented to the network are marked in the search space. Similarly, as all the neurons respond to every feature, the positions of the synapses are also marked in the same search space. The convention used is to give the features a light colour and they are designated by a capital letter, whereas the neurons are given a dark colour and are designated by a lower case letter.

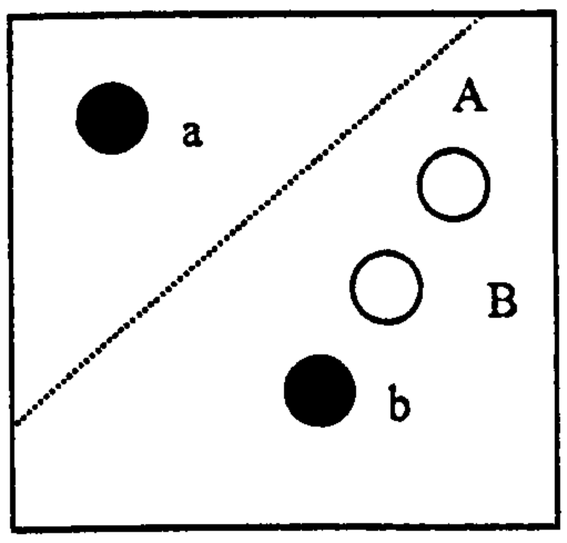


Figure 3.7 Synapse space

In this example the features are designated *A* and *B*, and the neurons *a* and *b*. The position of a feature relative to the neurons will determine which one of the two neurons will have

the greater output and will therefore classify the feature. Both neurons will have the same output when a feature is equidistant from each neuron. A locus of such points defines a straight line and is called the decision surface. This can be generalised to higher dimensions where the decision surface is a plane for neurons with three synapses and a hyper-plane for four or more synapses.

During learning the values for the synapse weights can change, so the neurons can *move* in the search space diagram. However, the locations of the features are fixed.

The arrangement of features and neurons in Figure 3.7, is such that both features will be classified by neuron *b*, with the larger output occurring when being presented with feature *B*. The synapse space diagram can be used to visually explain various features of learning.

As an example, if both features *A* and *B* are always presented, neuron *b* will classify feature *B*. The learning algorithm will update the synapse weight values for neuron *b*, causing it to move closer to feature *B*. On examination of the neuron output values, it will be found that neuron *b* will have a greater output than before when being presented with feature *B*. Hence it will be selected for learning and will move further towards feature *B*.

This can be described mathematically. The analysis can be simplified by considering a competition between neuron *a* learning feature *A*, and neuron *b* learning feature *B*.

Neuron b will be selected for learning when

$$|\mathbf{A}-\mathbf{a}|_k > |\mathbf{B}-\mathbf{b}|_k \qquad 3.41$$

At the start of the learning process, $k = 0$ giving

$$|\mathbf{A}-\mathbf{a}|_0 > |\mathbf{B}-\mathbf{b}|_0 \qquad 3.42$$

After teaching neuron *b*,

$$|\mathbf{B}-\mathbf{b}|_k > |\mathbf{B}-\mathbf{b}|_{k+1} \qquad 3.43$$

The initialisation conditions are such that Equation 3.42 is satisfied, causing neuron *b* to be taught and satisfying Equation 3.43. This in turn ensures that Equation 3.41 is always satisfied, therefore only neuron *b* will be taught.

As another example, the learning session starts with the same geometrical arrangement of neurons and features. Instead of both features being present, only one is present at any time starting with feature *B*. It would be desirable for both neurons to learn different features.

After the first presentation, neuron *b* classifies feature *B* and the synapse weight updating moves towards feature *B*. Now feature *A* is presented and it is entirely possible for neuron *b* to classify feature *A* and the teaching will move it towards feature *A*. No matter what feature is presented it will always be classified by neuron *b*. As far as the system is concerned, both features *A* and *B* are the same feature.

For the above scenario to happen the following conditions must be satisfied:

$$|\mathbf{A}-\mathbf{a}|_0 > |\mathbf{B}-\mathbf{b}|_0 \qquad 3.44$$

$$|\mathbf{A}-\mathbf{a}|_k > |\mathbf{B}-\mathbf{A}|_k \qquad 3.45$$

$$|\mathbf{B}-\mathbf{a}|_k > |\mathbf{B}-\mathbf{A}|_k \qquad 3.46$$

Equation 3.44 represents the initial conditions, Equations 3.45 and 3.46 represent the situation when a neuron has moved to a position coincident with a feature and that feature is not present at the start of another learning cycle. The positions of neuron *b* in the three equations are the extreme positions and if Equations 3.44 to 3.46 are valid, neuron *b* will dominate learning for all values of *k*.

The positions of the features are defined by the teaching set whereas the positions of the neurons are normally randomly determined. Therefore, in some circumstances all neurons will learn different features and sometimes a few neurons will tend to dominate.

One way of trying to give a more equitable learning strategy is to reduce the effectiveness of a neuron that has classified a feature and has been updated. One way of accomplishing this is to *tire* the neuron once it has been updated giving other neurons a better chance to learn the other features. This is similar to the process of habituation that is described in Chapter 2.

Figure 3.8 shows four possible learning sequences, the same convention as Figure 3.7 is used. However a dotted line is used to indicate the locus of the decision boundary. Two types of feature presentation can be illustrated:

- Each feature is alternately presented, starting with feature *B*

- Both features are always present

The following description is for each feature being alternately presented. The initial conditions are shown in Figure 3.8A with the decision boundary as a straight line with all

points equidistant between the neurons *a* and *b*. Neuron b will classify feature *B* and will move towards feature *B*. The decision boundary will be dragged towards the new position of neuron *b*. If feature *A* is always on the same side of the decision boundary as neuron *b* it will never be classified by neuron *a*.

However, if neuron *b* were made less effective at classifying, the decision boundary would change from a straight line to a closed convex curve. Initially the size of the boundary would be huge and only part of the curve would be visible. However, the bending of the boundary towards neuron b would give it a smaller area, or more generally a hypervolume. This is illustrated in Figure 3.8B.

Feature *A* is now presented instead of feature *B*, and because it is on the same side as neuron b it will be classified by neuron *b*. Neuron *b* will move towards feature *A* and will become more tired resulting in a smaller and more curved decision area. This is shown in Figure 3.8C.

For the next learning cycle, feature *B* is now presented instead of feature *A*. Neuron *b* classifies and moves towards feature *B* and becomes even more tired giving a smaller and more curved decision area. Feature *A* will be on the same side of the decision boundary as neuron *a*, this is illustrated in Feature 3.8D.

When feature *A* replaces feature *B*, it will be classified by neuron *a*, which will move towards feature *A* and will reposition the decision boundary. Neuron *a* will become tired whereas neuron *b* will become less tired. This results in the decision boundary opening out giving a larger area surrounding neuron *b*.

This sequence would form part of a much larger teaching sequence, and gives a realignment of the neurons so that neuron *a* will eventually learn feature *A* and neuron *b* will learn feature *B*.

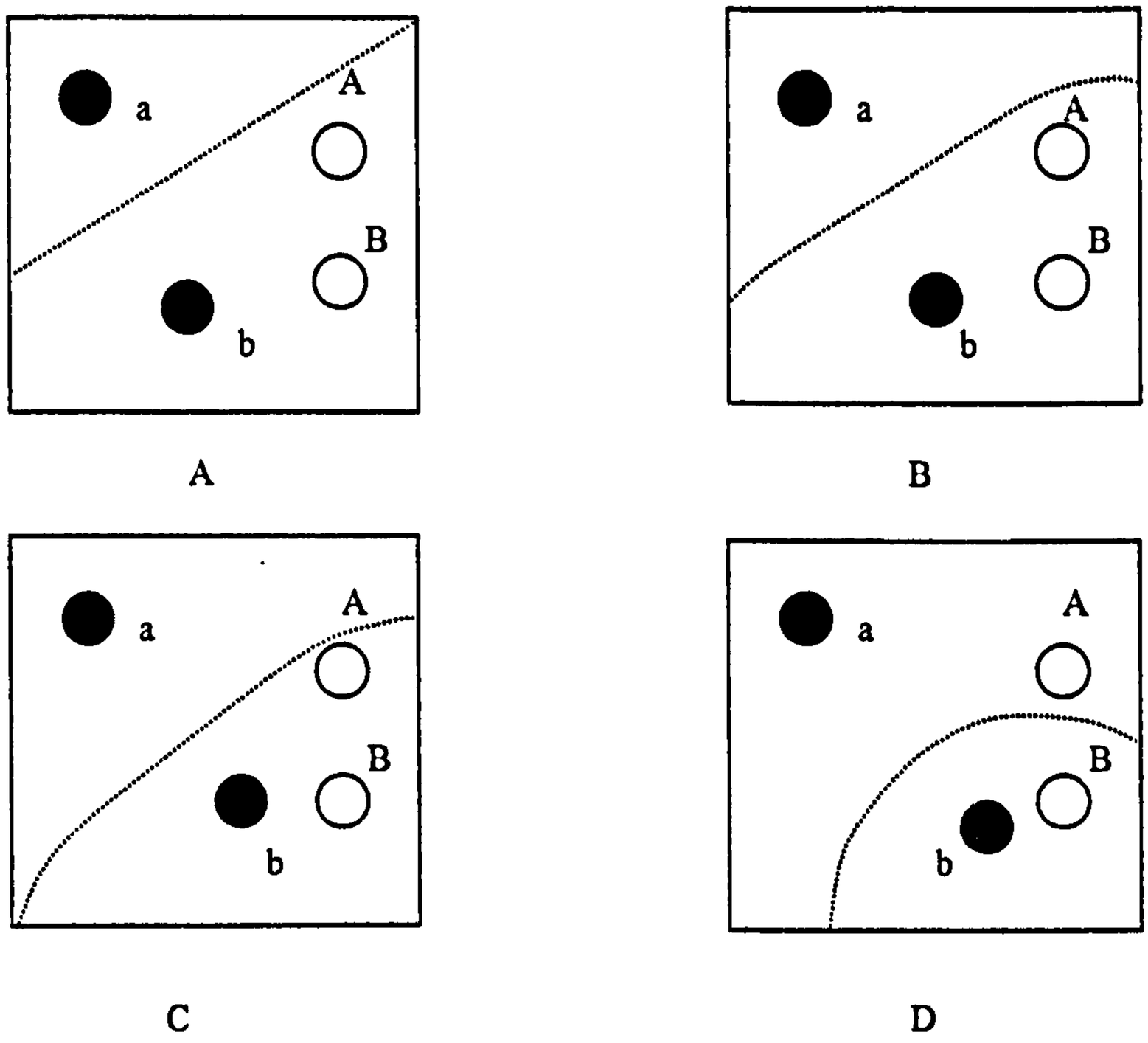


Figure 3.8 Tiring on alternating features

For the situation where both features are continuously presented, the initial conditions are shown in Figure 3.9A with the decision boundary as a straight line with all points equidistant between the neurons *a* and *b*. Neuron *b* will classify feature *B* and will move towards feature *B*. The decision boundary will be dragged towards the new position of neuron *b*. Tiring neuron *b* will curve the decision surface and reduce the effectiveness of the neuron. This is illustrated in Figure 3.9B.

For the next learning cycle neuron b will again classify feature B, and it will move towards feature B and the capture area/volume around will further decrease, as shown in Figure 3.9C. This situation continues until the effectiveness of neuron b has been reduced

allowing neuron a to classify the closest feature, in this case feature A is classified. See Figure 3.9D.

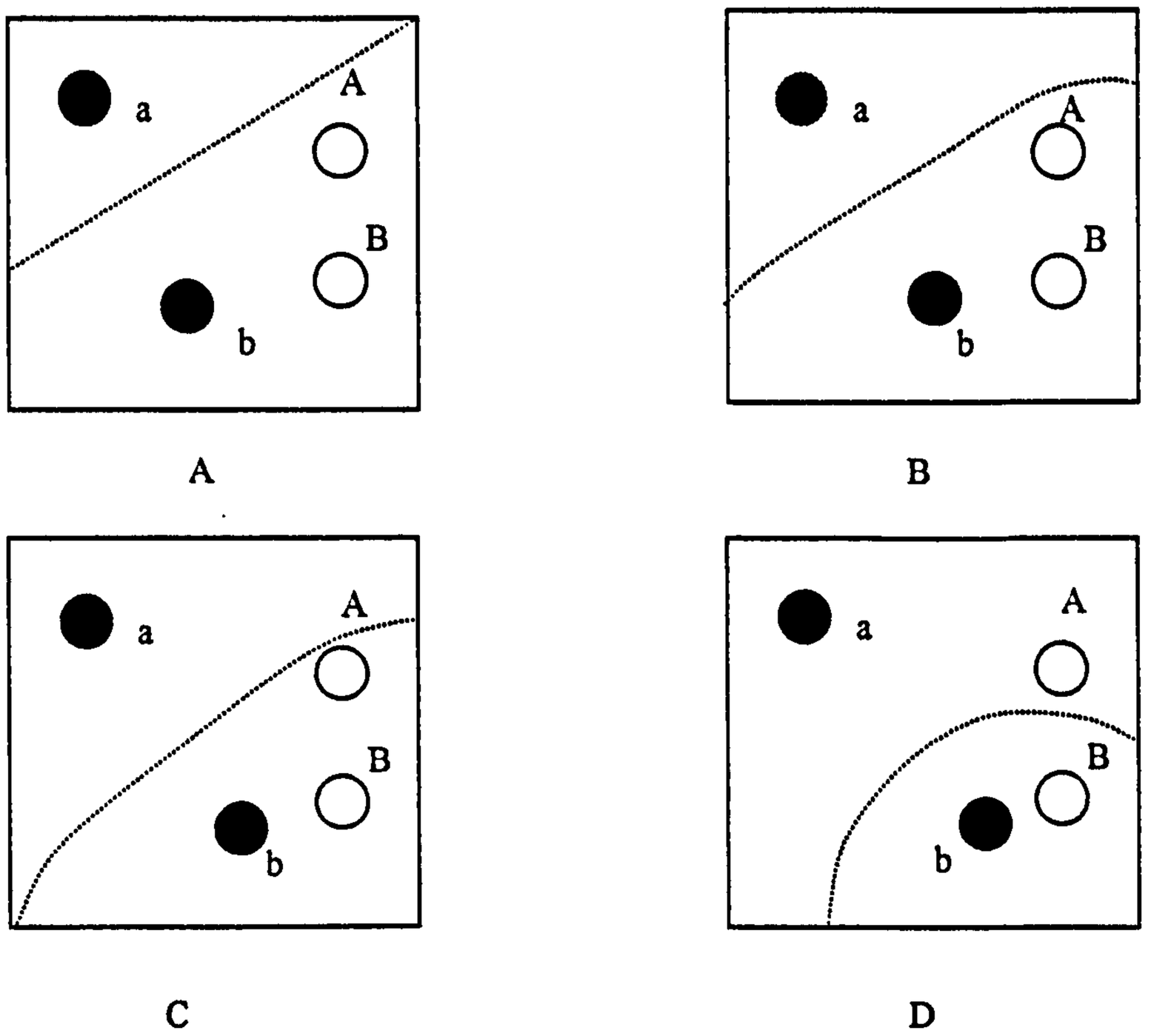


Figure 3.9 Tiring on continuous features

Having proposed the concept of tiring neurons after teaching, the implementation of such a system is discussed. The tiring process is purely concerned with learning and not pattern classification. Therefore, it should be separated from the normal operation of the neuron. This is particularly important in multi-layer networks.

The mechanism adopted was to provide a secondary output from the neuron called *the tired output*, which is related to the neuron output by Equation 3.47. A learning controller that determines which neurons will be taught only senses this tired output.

$$Output_{tired} = Output(T_{\max} - T_k) \qquad 3.47$$

$T_{\max}$ is the maximum tiredness and has value of 1

T_k is the instantaneous tiredness

With a tiredness of 0, the tired output is the same as the neuron output, and when the tiredness is 1, the neuron output is 0.

The effect of tiredness on the decision boundary between two neurons is shown in Appendix 1.

Two adaptation rules are used to determine the tiredness, an enhancing rule (Equation 3.48) and a reduction rule (Equation 3.49).

$$T_{k+1} = T_k + K_{te}(T_{\max} - T_k) \qquad 3.48$$

$$T_{k+1} = T_k - K_{tr}(T_k - T_{\min}) \qquad 3.49$$

$T_{\min}$ is the minimum tiredness and has a value of 0

K_{te} is the tiredness enhancement rate that has a range $0 \le K_{te} \le 1$

K_{tr} is the tiredness reduction rate that has a range $0 \le K_{tr} \le 1$

The tiredness enhancement rule is invoked as part of the neuron learning mechanism after it has classified a feature, and the tiredness reduction rule is invoked as part of the learning cycle for the neurons which are not being taught. In every learning cycle, one neuron should have its tiredness enhanced and the other neurons will have their tiredness reduced. The values for the tiredness coefficients are chosen so that $K_{te} >> K_{tr}$, rapidly tiring a neuron to give other neurons a chance of learning the features and letting tired neurons slowly recover. Ideally the tiredness - epoch graph should have a form similar to a sawtooth wave. This is shown in Figure 3.10.

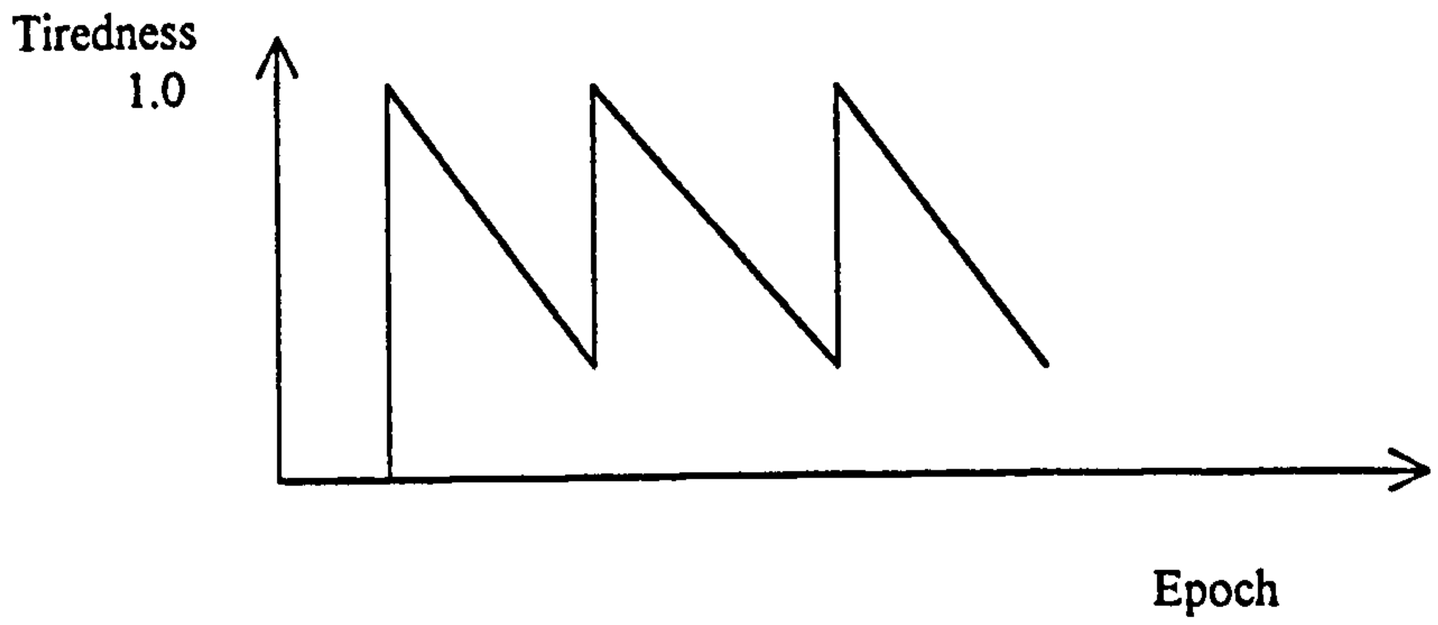


Figure 3.10 Ideal tiredness epoch graph

The form of the tiredness-enhancing graph is such that if a tired neuron still classifies, the tiredness can still increase to further reduce its effectiveness. Although the sawtooth wave form analogy is not correct with respect to the decay in tiredness value, it is a convenient term to use.

Different approaches can be used to prevent learning being dominated by a few neurons, for example, giving neurons a conscience [DeSieno 1988] or using backward inhibition [Hrycej 1989].

A conscience [DeSieno 1988] has been used for Kohonen self-organising maps [Kohonen 1984, Kohonen 1988]. In addition to the normal distance criterion for selecting a neuron,

an extra bias term (or conscience) has been added to favour neurons that have been infrequently selected for learning. The conventional method of determining which neuron is selected is given by Equation 3.50

$$|\mathbf{w}_i - \mathbf{x}|^2 < |\mathbf{w}_j - \mathbf{x}|^2 \qquad 3.50$$

This will select neuron *i* for updating.

The addition of the bias term modifies the neuron selecting condition:

$$|\mathbf{w}_i - \mathbf{x}|^2 - b_i < |\mathbf{w}_j - \mathbf{x}|^2 - b_j \qquad 3.51$$

The value for the bias term is determined from the proportion of the time that the neuron is selected for learning (p_i) and for a network with N competitive neurons it is given by Equation 3.52.

$$b_i = C\left(\frac{1}{N} - p_i\right) \qquad 3.52$$

C is the bias factor.

The value for p_i can be estimated by Equation 3.53.

$$p_{ik+1} = p_{ik} + B(y_i - p_{ik}) \qquad 3.53$$

A value for B is in the range $0 < B << 1$, and a typical value is 0.0001.

The value for y_i is the status of whether the neuron was selected for learning with $y_i = 1$ if it was selected and $y_i = 0$ if not.

The bias term becomes larger when the neuron has been successful at being selected for learning and small when unsuccessful. With the larger the bias term, the harder it is for the neuron to be selected. This bias term can act as a conscience, as if to indicate that the neuron is beginning to feel guilty, and prevents itself from being excessively selected for learning.

This system has a number of similarities with the concept of tiredness, apart from the conscience being a bias term and the tiredness being a multiplier. The main difference between the two systems is that different adaptation rates for increasing and decreasing the tiredness are used, whereas there is a constant rate for adapting the conscience.

Backward inhibition is used for perceptron type neurons. After a neuron has been taught, the input to the neurons in that layer is modified so that the previously taught neuron(s) give a smaller output. This is achieved by subtracting an orthogonal projection of the input vector **x** on the feature weight vector from the original input to the layer.

To suppress the activity of neuron j, the original input **x** is modified to **xx** by considering the weights (normalised to give a unit vector) of this neuron.

$$\mathbf{xx} = \mathbf{x} - \left(\frac{\mathbf{w}_j^T \mathbf{x}}{|\mathbf{w}_j|} \right) \mathbf{w}_j \qquad 3.54$$

The output of the neuron is the inner product of the weight and input vectors giving:

$$\mathbf{xx} = \mathbf{x} - y_j \mathbf{w}_j \qquad 3.55$$

The modified output can be determined by calculating the scalar product of the modified input and the feature weights.

$$yy_j = \mathbf{x}^T \mathbf{w}_j - y_j w_j^T \mathbf{w}_j = 0 \qquad 3.56$$

Considering Equation 3.55, the input is modified by the neuron's output multiplied by the weight vector. So the input is modified (inhibited) from the output from a higher layer. The inhibition signal direction is opposite (backward) to the direction of information processing in the network.

Even when using the tired output that is shown in Equation 3.47, it was found in practice that there are some situations where the learning is still dominated by a few neurons. The teaching algorithms were modified to try to improve on this situation.

The principle behind the modified teaching algorithm is to try to learn features, which cannot easily be learnt by the techniques previously mentioned because of the spatial arrangement of neurons and features. Because the neurons operate on a distance measure it is easy to determine which neuron is the closest to the input feature.

If a neuron which has the largest tired output is the closest to the input feature at that position, it is deemed to be the ideal candidate for teaching, and is taught to recognise this feature. This type of learning is called *updating*.

Failure to satisfy this distance criterion means that another neuron has already been taught, and is better positioned to learn this feature. If the other neuron had not been previously taught, it would have a larger tired output and would have been selected for learning. As the neuron with the largest tired output is not the closest to the input feature, it is not a good idea to automatically teach it the input feature.

All neurons with the same feature are tested throughout the capture area and the neuron with the next largest tired output is tested. If it is the closest, it is taught this feature, otherwise the search is repeated for the next largest tired output neuron. If the neuron learns a different feature, it is called *moving*. This is because the neuron (hopefully) moves to a more favourable position.

Failure of the neuron to learn by updating or by moving could be an indication that maybe there are no features that can be learnt, and the original feature identified was the most suitable. This is known as *clustering* because more than one neuron has clustered around a feature.

Clustering can be a valid strategy that can move neurons towards positions that are more favourable. A group of neurons could try to cluster around a group of features, and as the neurons approach the features, the features appear to separate and it is possible for the neurons to be taught by updating. This is illustrated in Figure 3.11.

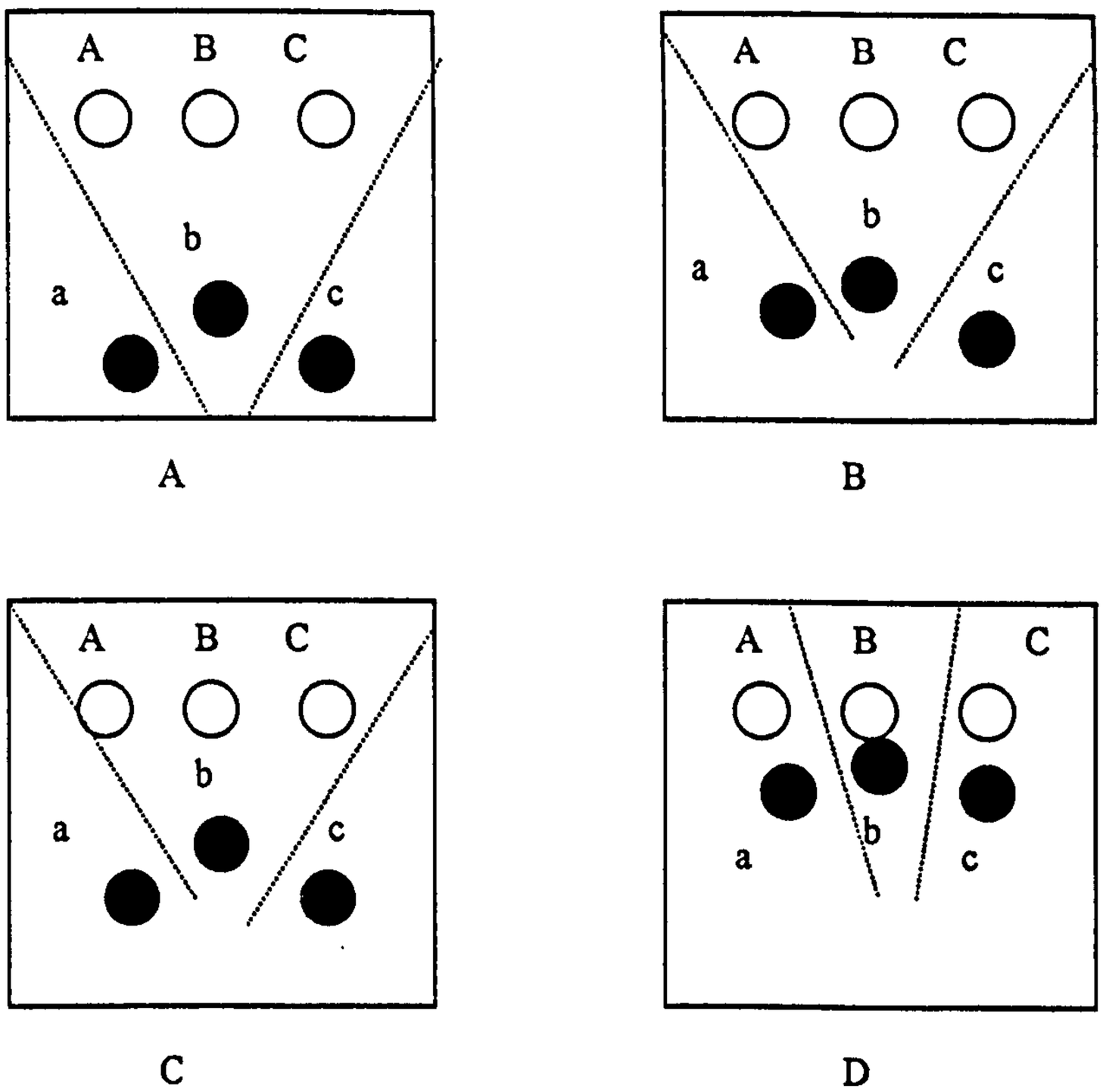


Figure 3.11 Changing from clustering to updating

Figure 3.11 shows three neurons with decision boundaries and three features. The initial arrangement as shown in Figure 3.11A is such that neuron *b* will classify all the features. This is illustrated by the decision boundaries enclosing all three features. Using the learning mechanisms described above, after a few learning epochs neuron *b* will learn by updating and neurons *a* and *c* will learn by clustering to give the arrangement shown in Figure 3.11B. This movement of the neurons towards the features will reduce the distance between the decision boundaries and the outermost features.

Further learning epochs will make the neurons cluster closer to the features as shown in Figure 3.11C, and the decision boundaries almost touch the features.

A point is eventually reached where the decision boundaries have moved so close to feature *B* that neuron *a* will classify feature *A*, and neuron *c* will classify feature *C*. Further learning will teach all the neurons by updating.

The effect of the *moving* teaching mechanism can also improve the performance of the learning process; this is illustrated in Figure 3.12.

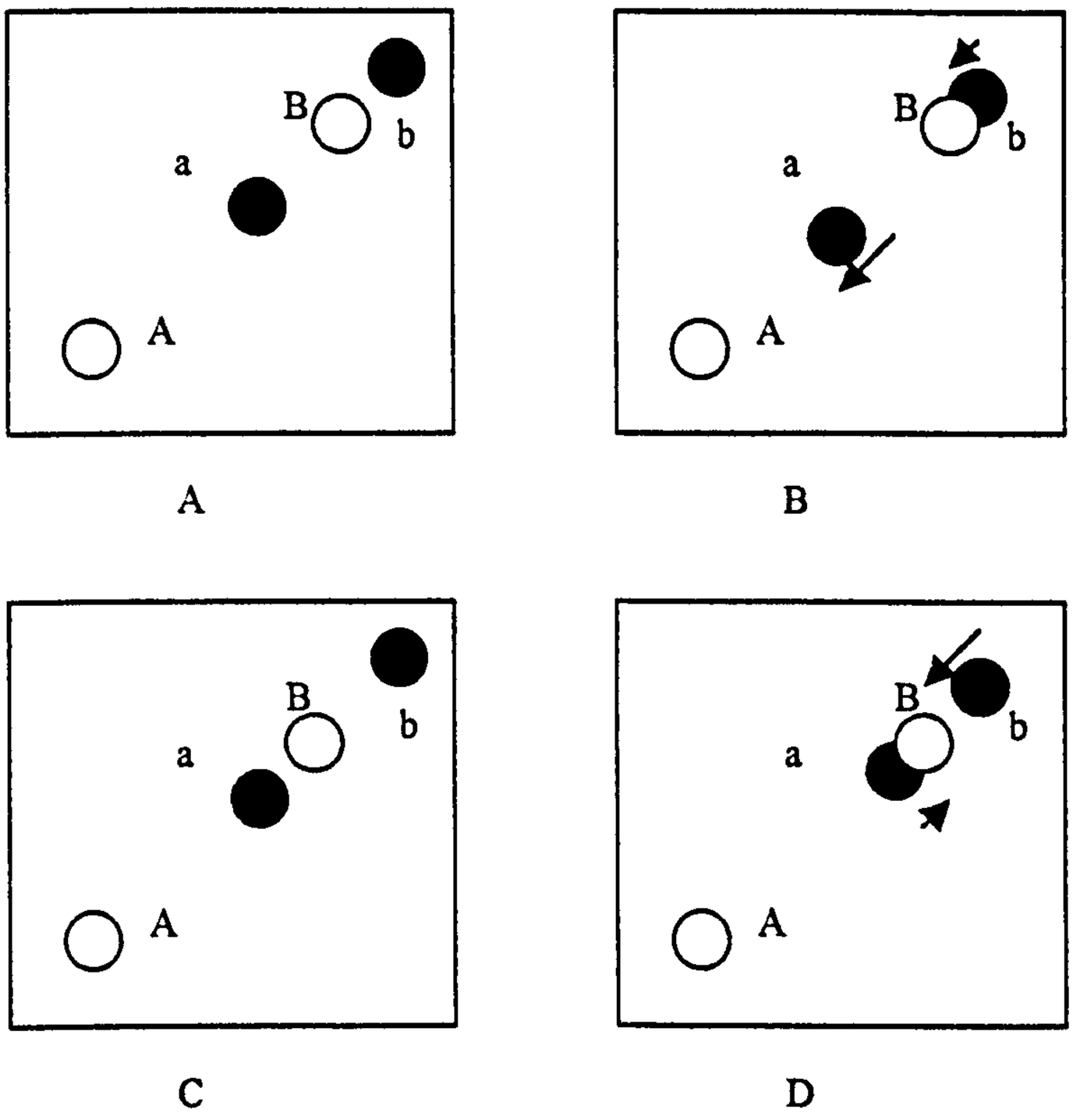


Figure 3.12 Learning by moving

Figure 3.12 shows two neurons (*a* and *b*) and two features (*A* and *B*), the goal is for each neuron to classify a different feature. For the initial arrangement shown in Figure 3.12A, both neurons *a* and *b* will exhibit a strong response to feature *B*, that is much greater than the response of neuron *a* to feature *A*. Normally the teaching would be dominated by feature *B*. During the initial learning stages, neuron *b* will be taught feature *B*. This is shown in Figure 3.12B.

As the learning progresses, neuron *b* becomes progressively more tired giving neuron *a* the opportunity to learn the features, and it will also classify feature *B*. When testing for the closest neuron to a feature it is discovered that neuron *b* is closer to a feature at the spatial location of the winning neuron *a*, and it is prevented from updating. All neurons of type *a* are searched, in the capture area, and a different neuron *a* will classify feature *A* and will also be the closest neuron. This neuron will be taught by moving, and the two neurons will separate.

If neuron *a* were originally closer to feature *B* than neuron *b*, learning by moving would not be invoked and both neurons would cluster around feature *B*. This is shown in Figures 3.12C and 3.12D.

3.6 IMPLEMENTATION

A software implementation was carried out on an Acorn Archimedes computer, the details of which are given in Appendix 2.

Appendix 3 gives the details of testing the network on binary input patterns. Two network topologies were tested: a single layered network and a triple layered network. The network was able to correctly classify the features in both cases with the single layered neuron giving better discrimination between the classified and non-classified features.

A summary of the results for the single layer network is shown in Table 3.2

Image Number	Image	Maximum Output
1	Number 0	1.000
2	Number 1	1.000
3	Number 2	1.000
4	Number 3	1.000
5	Number 4	1.000
6	Number 5	1.000
7	Number 6	1.000
8	Number 7	1.000
9	Number 8	1.000
10	Number 9	1.000
11	Noisy number 0	1.000
12	Noisy number 1	1.000
13	Noisy number 2	1.000
14	Noisy number 3	1.000
15	Noisy number 4	1.000
16	Noisy number 5	1.000
17	Noisy number 6	1.000
18	Noisy number 7	1.000
19	Noisy number 8	1.000
20	Noisy number 9	1.000
21	Shifted number 0	1.000
22	Shifted number 1	1.000
23	Shifted number 2	1.000

24	Shifted number 3	1.000
25	Shifted number 4	1.000
26	Shifted number 5	1.000
27	Shifted number 6	1.000
28	Shifted number 7	1.000
29	Shifted number 8	1.000
30	Shifted number 9	1.000
31	Background	1.000
32	Numbers 0, 1, 2, and 3	1.000

Table 3.2 Summary of the results for single layer network

A summary of the results for the triple layer network is shown in Table 3.3

Image Number	Image	Maximum Output
1	Number 0	1.000
2	Number 1	1.000
3	Number 2	1.000
4	Number 3	1.000
5	Number 4	0.965
6	Number 5	1.000
7	Number 6	1.000
8	Number 7	1.000
9	Number 8	1.000
10	Number 9	1.000
11	Noisy number 0	0.937
12	Noisy number 1	0.673
13	Noisy number 2	0.711
14	Noisy number 3	0.721
15	Noisy number 4	0.823
16	Noisy number 5	1.000
17	Noisy number 6	0.804
18	Noisy number 7	0.951
19	Noisy number 8	1.000
20	Noisy number 9	0.821
21	Shifted number 0	1.000
22	Shifted number 1	1.000
23	Shifted number 2	1.000

24	Shifted number 3	1.000
25	Shifted number 4	1.000
26	Shifted number 5	1.000
27	Shifted number 6	1.000
28	Shifted number 7	1.000
29	Shifted number 8	1.000
30	Shifted number 9	1.000
31	Background	1.000
32	Numbers 0, 1, 2, and 3	1.000

Table 3.3 Summary of the results for triple layer network

When learning, the single layered network was able to learn faster because of the reduced number of neurons.

The network has the ability to simultaneously classify multiple features and to preserve their spatial relationships.

3.7 DISCUSSION

The classification performance of two network configurations were tested under the following conditions:

- Classification of the teaching set.
- Classification of noisy images.
- Classification of translated images.
- Classification of multiple features.

The single layer network was superior to the triple layer network, it was capable of better discrimination and was less susceptible to noise. The triple layer network found it difficult to discriminate between the numbers 5 and 6 and between 3 and 8. On analysis of the results, it was found that the inferior results for the triple layer were caused by two factors:

- Tolerance

  The tolerance was compounded through three layers and reduced the discrimination capabilities.

- Smaller receptive field

  The smaller receptive field meant that noise caused by one or two pixels had a much larger effect than for the single layer network with the larger receptive field.

In addition to the classification performance, the single layer network was faster at learning and classifying because of the smaller number of neurons.

4.0 A RETINAL MODEL AND GREY SCALE CLASSIFICATION

4.1 CHAPTER CONTENTS

The research described in this chapter tests the shift invariant neural network developed in Chapter 3 with grey scale inputs. One way of generating grey scale inputs is to process binary inputs, using an artificial retina. The effect of the retina on changes in image brightness could also be examined. Other work on an artificial retina has produced a two layer retinal system that has edge detection and image segmentation properties [Yi et al. 1997].

The author has developed an existing mathematical formulation for a retina into a time-independent matrix model and the work is described in this chapter. Similarly, the author has developed an existing mathematical method for solving the large matrix equations into a computer algorithm that was used to predict the retina output.

4.2 GREY SCALE CLASSIFICATION

Chapter 3 developed the ideas for a shift invariant neural network. It was tested with binary inputs, that is inputs with levels of either 0 or 1. Further tests must be carried out to determine the performance of the network with grey scale images, that is values that can lie in a continuous range, in this case between 0 and 1. It is expected that this classification will be more difficult to achieve because the neuronal equations were developed to give their best performance with binary images.

Although grey scale images can easily be constructed, it was decided to use the concept of an artificial retina. This allowed the effect of changed in brightness level to be examined.

4.3 A MODEL FOR THE RETINA

Chapter 2 gave a simplified description for the structure and behaviour of a mammalian retina. The good performance of the retina when performing feature extraction under a range of lighting conditions has encouraged further study. One offshoot of this research has been the construction of an artificial retina [Mead and Mahowald 1988 and Mahowald and Mead 1991], which can provide an output that is analogous to the output from the bipolar cell in the vertebrate retina.

Figure 2.3 shows the structure of a mammalian retina. Considering its behaviour from an Engineering viewpoint, the following properties can be identified:

- The photoreceptor has an output that is proportional to the logarithm of the light intensity.
- The photoreceptor output is spatially and temporally averaged by the horizontal cells.
- The bipolar cells have an output that is proportional to the difference between the photoreceptor output and the horizontal cell output.

The photoreceptor of the artificial retina [Mead and Mahowald 1988] was constructed from a vertical bipolar transistor, the output of which is fed into a circuit with an exponential current-voltage relationship. Two diode-connected MOS transistors arranged in series provide this exponential characteristic.

The horizontal cells were modelled by a hexagonal network of resistive elements, which used transistors to provide the resistance. Each node of the network was driven by the

photoreceptor element via a conductance. The differential output across this conductance is used to drive a differential amplifier.

The circuit diagram for part of the retina is shown in Figure 4.1

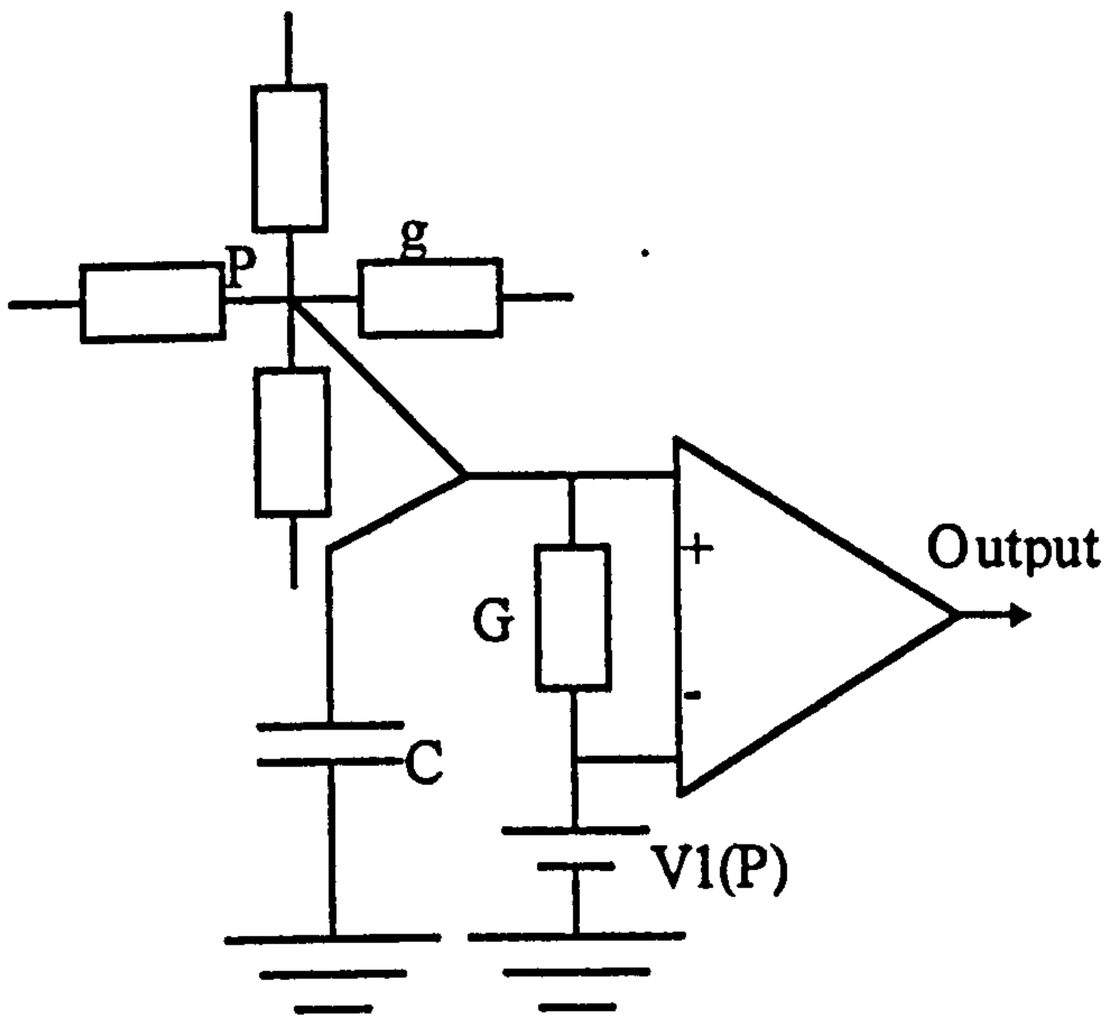


Figure 4.1 Circuit diagram for part of the retina

The capacitance of the model is formed by the substrate.

An analysis of this retina has been carried out [Taylor 1990] in which the nodal voltage equations are linearised. The same nodal voltage equation is given by Equation 4.1

$$G[V_1(P) - V_p] - CV_p = g[NV_p - \sum V_q] \qquad 4.1$$

Where

$V_1(P)$ = Photoreceptor voltage at position P

V_p = Node voltage at position P

V_q = Node voltage at position Q

G = Receptor conductance

g = Matrix conductance

N = Number of neighbours

C = Substrate capacitance

Apart from edges and corners of the retina, the hexagonal arrangement of the matrix will give 6 neighbours for the each node of the artificial retina.

The substrate capacitance gives the model time-dependence; setting this value to zero gives the time-independent equation shown in Equation 4.2.

$$G[V_1(P) - V_p] = g[NV_p - \sum V_q] \qquad 4.2$$

For a finite reticule matrix, there will be the following number of neighbouring nodes:

- 2 neighbours at a corner
- 3 neighbours at an edge
- 4 neighbours elsewhere

To simplify the following equations, the *(P)* term will be dropped.

The retina node labelling system used is shown in Figure 4.2.

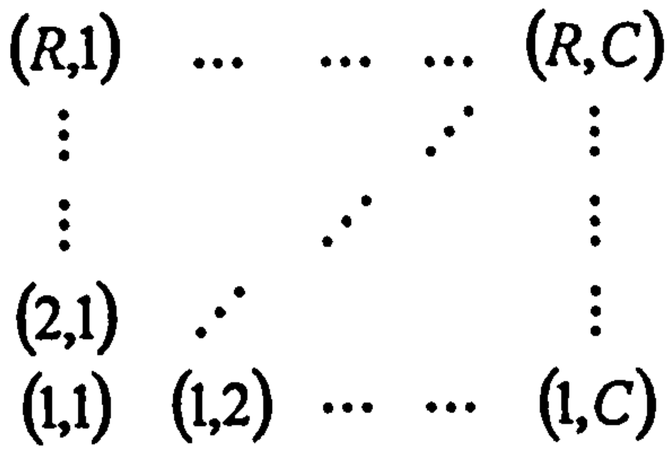


Figure 4.2 Node labelling system

Where

R is the number of rows

C is the number of columns

As an example of how the terms are constructed, equation 4.2 is applied at node 1,1.

$$G[V_1(1,1) - V(1,1)] = g[2V(1,1) - V(2,1)] \qquad 4.3$$

Rearranging gives

$$GV_1(1,1) = (2g + G)V(1,1) - gV(1,2) - gV(2,1) \qquad 4.4$$

Finally giving

$$V_1(1,1) = \frac{(2g + G)V(1,1)}{G} - \frac{gV(1,2)}{G} - \frac{gV(2,1)}{G} \qquad 4.5$$

Similar equations can be constructed for all nodes.

Considering the nodal voltage equations at each node and expressing the equations in matrix form, gives the relationship between the photoreceptor voltage and the nodal voltage as shown in Equation 4.6.

$$BV = V_1 \qquad 4.6$$

Where

B is the $RC \times RC$ matrix of coefficients made up from conductances

V is a RC column vector of nodal voltages

V_1 is a RC column vector of photoreceptor voltages

The terms in the coefficients matrix are arranged in a highly structured manner. As an example Equation 4.5 shows the terms of Equation 4.6 for a 3 × 3 retina.

$$\begin{bmatrix}
\frac{2g+G}{G} & -\frac{g}{G} & 0 & -\frac{g}{G} & 0 & 0 & 0 & 0 & 0 \\
-\frac{g}{G} & \frac{3g+G}{G} & -\frac{g}{G} & 0 & -\frac{g}{G} & 0 & 0 & 0 & 0 \\
0 & -\frac{g}{G} & \frac{2g+G}{G} & 0 & 0 & -\frac{g}{G} & 0 & 0 & 0 \\
-\frac{g}{G} & 0 & 0 & \frac{3g+G}{G} & -\frac{g}{G} & 0 & -\frac{g}{G} & 0 & 0 \\
0 & -\frac{g}{G} & 0 & -\frac{g}{G} & \frac{4g+G}{G} & -\frac{g}{G} & 0 & -\frac{g}{G} & 0 \\
0 & 0 & -\frac{g}{G} & 0 & -\frac{g}{G} & \frac{3g+G}{G} & 0 & 0 & -\frac{g}{G} \\
0 & 0 & 0 & -\frac{g}{G} & 0 & 0 & \frac{2g+G}{G} & -\frac{g}{G} & 0 \\
0 & 0 & 0 & 0 & -\frac{g}{G} & 0 & -\frac{g}{G} & \frac{3g+G}{G} & -\frac{g}{G} \\
0 & 0 & 0 & 0 & 0 & -\frac{g}{G} & 0 & -\frac{g}{G} & \frac{2g+G}{G}
\end{bmatrix}
\begin{bmatrix} V(1,1) \\ V(1,2) \\ V(1,3) \\ V(2,1) \\ V(2,2) \\ V(2,3) \\ V(3,1) \\ V(3,2) \\ V(3,3) \end{bmatrix} =
\begin{bmatrix} V_1(1,1) \\ V_1(1,2) \\ V_1(1,3) \\ V_1(2,1) \\ V_1(2,2) \\ V_1(2,3) \\ V_1(3,1) \\ V_1(3,2) \\ V_1(3,3) \end{bmatrix} \qquad 4.7$$

Comparing Equation 4.3 with Equation 4.5, the following can be deduced:

The $-\frac{g}{G}$ terms are caused by the effect of the voltage at the adjacent nodes.

The $\frac{2g+G}{G}$, $\frac{3g+G}{G}$ and $\frac{4g+G}{G}$ terms are due to the effect of the photoreceptor and node voltages.

The number of $-\frac{g}{G}$ terms will be equal to the coefficient in the $\frac{2g+G}{G}$, $\frac{3g+G}{G}$ and $\frac{4g+G}{G}$ terms.

For a square resistance network there will be a maximum of 5 non-zero terms in each row or column (node plus up to 4 adjacent nodes).

Further deductions can also be made;

- The matrix will be symmetrical.
- The matrix will be sparse.
- The matrix will be banded.

When constructing the terms of the coefficients a mapping function is required to map the coefficients' matrix terms to the retina matrix terms. Once this has been established, the form of the matrix can be considered to be constructed from the effects of the photoreceptor voltages and adjacent node voltages (diagonal terms) and the effect of the adjacent node voltages (non-diagonal terms). These effects can be considered separately, the following algorithm being used to construct the matrix:

IF (matrix row = matrix column)

BEGIN

nodes = 0;

IF NOT retina left side **THEN** increment nodes

IF NOT retina right side **THEN** increment nodes

IF NOT retina bottom **THEN** increment nodes

IF NOT retina top **THEN** increment nodes

$$\text{value} = \frac{(nodes * g + G)}{G}$$

END

ELSE

BEGIN

IF (adjacent node above)

OR

(adjacent node below)

OR

(adjacent node to left)

OR

(adjacent node to right)

BEGIN

$$\text{value} = -\frac{g}{G}$$

END

ELSE

BEGIN

value = 0

END

END

The photoreceptor has a logarithmic characteristic and the exact form of this characteristic depends on the range of synapse input values used throughout the network. In Chapter 3, the synapse input values were in the range 0.0 to 1.0. For compatibility, this same range of values was processed and produced by the retina.

The model used for the photoreceptor is given in Equation 4.8.

$$V_1 = \log_{10}(9.0i + 1.0) \qquad 4.8$$

When i is 0 V_1 is 0, and when i is 1.0 V_1 is also 1.0. Because the function is monotonically increasing, the range of values for V_1 is also 0 to 1.0.

The output from the retina is proportional to the difference between the retina output and the matrix node. This value can be positive or negative. To restrict the range of output values to between 0 and 1.0 Equation 4.9 was used.

$$O = A(V_1 - V_P) + 0.5 \qquad 4.9$$

where A is the amplifier gain

4.4 CALCULATING THE RETINA OUTPUT

Calculating the photoreceptor output and the retina output is very straightforward. Problems arise when calculating the voltages in the reticule matrix because the number of nodes is proportional to the square of the size of the retina. A small retina of size 16 x 16 will have 256 nodes, giving a coefficients matrix of 256 x 256. A larger retina of size 32 x 32 will have 1024 nodes, giving a much larger coefficients matrix of 1024 x 1024.

It was previously deduced that the coefficients matrix is both symmetrical and sparse. One procedure for iteratively solving simultaneous equations is to use the conjugate gradients method (CGM) [Hestenes and Stiefel 1952], which can be used for positive definite systems. This method was improved by Lanczos [Lanczos 1950], in which the matrix is not

required to be definite. One further refinement of this method, which can efficiently solve symmetric and indefinite matrices, has been developed by [Paige and Saunders1975]. This method has advantages when solving large and sparse matrices.

An algorithm using this method has been developed to calculate the retina output. A dual system for identifying equations will be used when explaining how the algorithm works, the normal notation used throughout the chapter, and the number of the equation used in the paper but prefixed by P to indicate Paige and Saunders.

The explanation of the methodology uses the same convention used by Paige and Saunders;

matrices - upper case italic

vectors - lower case italic

scalars - lower case Greek

The object of the method is to solve for x in

$$Ax = b \qquad \text{4.10, P1.1}$$

Where A is an $n \times n$ real symmetric matrix.

Introducing a residual vector r, Equation 4.9 can be extended to give

$$r + Ax = b, Ar = 0 \qquad \text{4.11, P2.1}$$

Approximation to x of the form $V_k y, V_k \equiv \left[v_1, v_2, \cdots, v_k\right]$ can be made, where the v_k is a set of linearly independent vectors produced by an iterative process.

Unfortunately Paige and Saunders did not give enough detail to directly implement an algorithm; the additional working is shown below.

For iteration step k =1

From P3.3

$$T_k = [\alpha_k] \qquad 4.12, \text{P3.3}$$

From P5.5

$$T_k \equiv \overline{L}_k \equiv [\overline{\gamma}_k] \qquad 4.13, \text{P5.5}$$

From 4.12 and 4.13

$$\overline{\gamma}_k = \alpha_k \qquad 4.14$$

From P3.3

$$T_{k+1} \equiv \begin{bmatrix} \alpha_k & \beta_{k+1} \\ \beta_{k+1} & \alpha_{k+1} \end{bmatrix} \qquad 4.15, \text{P3.3}$$

From P5.4

$$Q_{k,k+1} \equiv \begin{bmatrix} \cos\theta_k & \sin\theta_k \\ \sin\theta_k & -\cos\theta_k \end{bmatrix} \qquad 4.16, \text{P5.4}$$

From P5.5

$$T_{k+1}Q_{k,k+1} \equiv \overline{L}_{k+1} \equiv \begin{bmatrix} \gamma_k & 0 \\ \delta_{k+1} & \overline{\gamma}_{k+1} \end{bmatrix} \qquad 4.17, \text{P5.5}$$

Combining 4.12, 4.13 and 4.14

$$\begin{bmatrix} \gamma_k & 0 \\ \delta_{k+1} & \bar{\gamma}_{k+1} \end{bmatrix} = \begin{bmatrix} \alpha_k & \beta_{k+1} \\ \beta_{k+1} & \alpha_{k+1} \end{bmatrix} \begin{bmatrix} \cos\theta_k & \sin\theta_k \\ \sin\theta_k & -\cos\theta_k \end{bmatrix} \qquad 4.18$$

Multiplying the matrices and solving gives:

$$\theta_k = \tan^{-1}\left(\frac{\beta_{k+1}}{\alpha_k}\right) \qquad 4.19$$

$$\delta_{k+1} = \beta_{k+1}\cos\theta_k + \alpha_{k+1}\sin\theta_k \qquad 4.20$$

$$\bar{\gamma}_{k+1} = \beta_{k+1}\sin\theta_k - \alpha_{k+1}\cos\theta_k \qquad 4.21$$

It can be deduced from equation P3.3 that

$$e_1 = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \qquad 4.22$$

Equation P5.4 gives:

$$\bar{L}_k \bar{z}_k = \beta_1 e_1 \qquad 4.23, \text{P5.4}$$

Considering the terms in equations 4.22, P4.3 and P4.5 remembering that k = 1, the matrix and vectors reduce to scalars and equation 4.23 can be simplified giving:

$$\bar{\gamma}_k \bar{\zeta}_k = \beta_1 1 \qquad 4.24$$

Solving for $\bar{\zeta}_k$ gives;

$$\bar{\zeta}_k = \frac{\beta_1}{\bar{\gamma}_k} \qquad 4.25$$

For iteration step k = 2

From P5.5

$$\bar{L}_{k+1} = \begin{bmatrix} \gamma_{k-1} & 0 & 0 \\ \delta_k & \gamma_k & 0 \\ \varepsilon_{k+1} & \delta_{k+1} & \bar{\gamma}_{k+1} \end{bmatrix} \qquad 4.26, \text{P5.5}$$

From P3.2

$$T_{k+1} = \begin{bmatrix} \alpha_{k-1} & \beta_k & 0 \\ \beta_k & \alpha_k & \beta_{k+1} \\ 0 & \beta_{k+1} & \alpha_{k+1} \end{bmatrix} \qquad 4.27, \text{P3.2}$$

From P5.4

$$Q_{k-1,k} = \begin{bmatrix} \cos\theta_k & \sin\theta_k & 0 \\ \sin\theta_k & -\cos\theta_k & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad 4.28, \text{P5.4}$$

$$Q_{k,k+1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_k & \sin\theta_k \\ 0 & \sin\theta_k & -\cos\theta_k \end{bmatrix} \qquad 4.29, \text{P5.4}$$

Solving for $\overline{L}_{k+1} = T_{k+1}Q_{k-1,k}Q_{k,k+1}$ and equating terms gives:

$$\varepsilon_{k+1} = \beta_{k+1}\sin\theta_{k-1} \qquad 4.30$$

$$\delta_{k+1} = -\beta_{k+1}\cos\theta_{k-1}\cos\theta_k + \alpha_{k+1}\sin\theta_k \qquad 4.31$$

$$\overline{\gamma}_{k+1} = -\beta_{k+1}\cos\theta_{k-1}\sin\theta_k - \alpha_{k+1}\cos\theta_k \qquad 4.32$$

From P5.4

$$\begin{bmatrix}\gamma_{k-1} & 0\\ \delta_k & \overline{\gamma}_k\end{bmatrix}\begin{bmatrix}\zeta_{k-1}\\ \overline{\zeta}_k\end{bmatrix} = \beta_1\begin{bmatrix}1\\ 0\end{bmatrix} \qquad 4.33$$

Giving

$$\overline{\zeta}_k = -\frac{\delta_k\zeta_{k-1}}{\overline{\gamma}_k} \qquad 4.34$$

For iteration step k > 2.

For iteration steps that are greater that two the situation conforms to a regular pattern. This relationship is developed by considering the most recent terms in Equations P3.3 and P5.5, that is the terms at the bottom right of the matrices. Using this technique the older bottom right terms of the $Q_{k-1,k}$ matrices are reduced to a unit matrix and have no effect.

The relationships are developed by considering sub matrices made up from the bottom right 4×4 terms. To distinguish these matrices from their full form they are denoted by the symbol ~.

from P5.4

$$\tilde{Q}_{k,k+1} = \begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & \cos\theta_k & \sin\theta_k\\ 0 & 0 & \sin\theta_k & -\cos\theta_k\end{bmatrix} \qquad 4.35$$

$$\tilde{Q}_{k-1,k} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta_{k-1} & \sin\theta_{k-1} & 0 \\ 0 & \sin\theta_{k-1} & -\cos\theta_{k-1} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad 4.36$$

$$\tilde{Q}_{k-2,k-1} = \begin{bmatrix} \cos\theta_{k-2} & \sin\theta_{k-2} & 0 & 0 \\ \sin\theta_{k-2} & -\cos\theta_{k-2} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad 4.37$$

$$\tilde{Q}_{k-3,k-2} = \begin{bmatrix} -\cos\theta_{k-3} & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad 4.38$$

Older terms are unit matrices

From P3.3

$$\tilde{T}_{k+1} = \begin{bmatrix} \alpha_{k-2} & \beta_{k-1} & 0 & 0 \\ \beta_{k-1} & \alpha_{k-1} & \beta_k & 0 \\ 0 & \beta_k & \alpha_k & \beta_{k+1} \\ 0 & 0 & \beta_{k+1} & \alpha_{k+1} \end{bmatrix} \qquad 4.39$$

Considering the lower triangular matrix

$$\tilde{\tilde{L}}_{k+1} = \begin{bmatrix} \gamma_{k-2} & 0 & 0 & 0 \\ \delta_{k-1} & \gamma_{k-1} & 0 & 0 \\ \varepsilon_k & \delta_k & \gamma_k & 0 \\ 0 & \varepsilon_{k+1} & \delta_{k+1} & \bar{\gamma}_{k+1} \end{bmatrix} \qquad 4.40$$

Calculating $\tilde{T}_{k+1}\tilde{Q}_{k-3,k-2}\tilde{Q}_{k-2,k-1}\tilde{Q}_{k-1,k}\tilde{Q}_{k,k+1}$ and equating the bottom line of this product with the bottom line of $\tilde{\tilde{L}}_{k+1}$ gives:

$$\varepsilon_{k+1} = \beta_{k+1}\sin\theta_{k-1} \qquad 4.41$$

$$\delta_{k+1} = -\beta_{k+1}\cos\theta_{k-1}\cos\theta_k + \alpha_{k+1}\sin\theta_k \qquad 4.42$$

$$\bar{\gamma}_{k+1} = -\beta_{k+1}\cos\theta_{k-1}\sin\theta_k - \alpha_{k+1}\cos\theta_k \qquad 4.43$$

Expressed in terms of iteration steps, the equations 4.41 to 4.43 are the same as equations 4.30 to 4.32.

Similarly considering the bottom right four rows and columns of the matrix and vectors in P5.4 gives

$$\begin{bmatrix} \gamma_{k-3} & 0 & 0 & 0 \\ \delta_{k-2} & \gamma_{k-2} & 0 & 0 \\ \varepsilon_{k-1} & \delta_{k-1} & \gamma_{k-1} & 0 \\ 0 & \varepsilon_k & \delta_k & \bar{\gamma}_k \end{bmatrix} \begin{bmatrix} \zeta_{k-3} \\ \zeta_{k-2} \\ \zeta_{k-1} \\ \bar{\zeta}_k \end{bmatrix} = \beta_1 \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \qquad 4.44$$

Multiplying the sub matrix and the vector, and equating terms gives

$$\bar{\zeta}_k = -\frac{(\varepsilon_k\zeta_{k-2} + \delta_k\zeta_{k-1})}{\bar{\gamma}_k} \qquad 4.45$$

An example of an algorithm based on Paige and Saunder's work for solving simultaneous equations is given below. The user supplies the coefficients matrix, the receptor voltages and the target residual that determines the accuracy of the calculation.

Allocate memory for the temporary arrays

Comment - initialise variables

$k = 1$

$$v_{k-1} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

$$x_{k-1} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

$\beta_k = |b|$ From P3.1

$\beta_1 = \beta_k$

Comment - estimate the residual

$|r_k| = \beta_1$

$v_k = \frac{b}{\beta_k}$ From P3.1

Comment - calculate Av_k

$\alpha_k = v_k^T A v_k$ From P3.2

WHILE ($|r_k|$ > target residual)

BEGIN

Comment - iterate using Lanczos algorithm

$\beta_{k+1} v_{k+1} = Av_k - \alpha_k v_k - \beta_k v_{k-1}$ From P3.2

$\beta_{k+1} = |\beta_{k+1} v_{k+1}|$ From P3.2

$v_{k+1} = \frac{\beta_{k+1} v_{k+1}}{\beta_{k+1}}$ From P3.2

Calculate Av_{k+1} From P3.2

$\alpha_{k+1} = v_{k+1} A v_{k+1}$ From P3.2

IF (k=1)

BEGIN

$$\theta_k = \tan^{-1}\left(\frac{\beta_{k+1}}{\alpha_k}\right)$$ From 4.19

Calculate $\sin\theta_k$

Calculate $\cos\theta_k$

$$\delta_{k+1} = \beta_{k+1}\cos\theta_k + \alpha_{k+1}\sin\theta_k$$ From 4.20

$$\bar{\gamma}_{k+1} = \beta_{k+1}\sin\theta_k - \alpha_{k+1}\cos\theta_k$$ From 4.21

$$\bar{\gamma}_k = \alpha_k$$ From 4.14

$$\bar{\zeta}_k = \frac{\beta_1}{\bar{\gamma}_k}$$ From 4.25

$$\bar{w}_k = v_k$$ From P5.9

END

ELSE

BEGIN

$$\gamma_k = \sqrt{\bar{\gamma}_k^2 + \beta_{k+1}^2}$$ From P5.6

IF ($\bar{\gamma}_k < 0$)

THEN

BEGIN

Comment - correct sign due to ambiguity of √ function

$\gamma_k = -\gamma_k$

END

$$\cos\theta_k = \frac{\bar{\gamma}_k}{\alpha_k}$$ From P5.6

$$\sin\theta_k = \frac{\beta_{k+1}}{\gamma_k}$$ From P5.6

$$\varepsilon_{k+1} = \beta_{k+1} \sin\theta_{k-1}$$ From 4.30, 4.41

$$\delta_{k+1} = -\beta_{k+1} \cos\theta_{k-1} \cos\theta_k + \alpha_{k+1} \sin\theta_k$$ From 4.31, 4.42

$$\bar{\gamma}_{k+1} = -\beta_{k+1} \cos\theta_{k-1} \sin\theta_k - \alpha_{k+1} \cos\theta_k$$ From 4.32, 4.43

IF $(k = 2)$

BEGIN

$$\bar{\zeta}_k = -\frac{\delta_k \zeta_{k-1}}{\bar{\gamma}_k}$$ From 4.34

END

ELSE

BEGIN

$$\bar{\zeta}_k = -\frac{(\varepsilon_k \zeta_{k-2} + \delta_k \zeta_{k-1})}{\bar{\gamma}_k}$$ From 4.45

END

END

$$\zeta_k = \bar{\zeta}_k \cos\theta_k$$ From P5.8

$$w_k = \bar{w}_k \cos\theta_k + v_{k+1} \sin\theta_k$$ From P5.9

$$\beta_1 \sin\theta_1 \sin\theta_2 \cdots \sin\theta_k = \beta_1 \sin\theta_1 \sin\theta_2 \cdots \sin\theta_{k-1} \times \sin\theta_k$$ From P5.16

Comment - update estimate for solution vector

$$x_k = x_{k-1} + w_k \zeta_k$$ From P5.10

$$r_k = -\frac{\beta_1 \sin\theta_1 \sin\theta_2 \cdots \sin\theta_k v_{k+1}}{\cos\theta_k}$$ From P5.16

Calculate $|r_k|$

$$\beta_k = \beta_{k+1}$$

$$\bar{\gamma}_k = \bar{\gamma}_{k+1}$$

$$\varepsilon_k = \varepsilon_{k+1}$$

$$\delta_k = \delta_{k+1}$$

$$\sin\theta_{k-1} = \sin\theta_k$$

$$\cos\theta_{k-1} = \cos\theta_k$$

$$\zeta_{k-2} = \zeta_{k-1}$$

$$\zeta_{k-1} = \zeta_k$$

$$\alpha_{k-1} = \alpha_k$$

$$\alpha_k = \alpha_{k+1}$$

$$\alpha_k = \alpha_{k+1}$$

$$v_{k-1} = v_k$$

$$v_k = v_{k+1}$$

$$\overline{w}_k = \overline{w}_{k+1}$$

$$w_{k-1} = w_k$$

$$k = k + 1$$

END

Free memory

4.5 ALTERNATIVE PHOTORECEPTOR MODELLING

An alternative model for the photoreceptor was also used to improve the invariance to image brightness. Instead of using the relationship described by Equation 4.8 the model shown in Equation 4.46 was used.

$$V_1 = \log_{10}(10i) \qquad 4.46$$

Although this is a simpler model than the model described by Equation 4.8 it is mathematically less robust because $V_1 \to -\infty$ as $i \to 0$. However, from the image

processing point of view it is a much better model, assuming that there are two brightness levels in the image and all pixels apart from one have the value V_B and the remaining pixel has the value V_A. The node voltage will approximate to the voltage given by Equation 4.46. Using these values in Equation 4.9 gives:

$$O = A(\log_{10}(10V_A) - \log_{10}(10V_B)) + 0.5 \qquad 4.47$$

or

$$O = A\log_{10}\left(\frac{V_A}{V_B}\right) + 0.5 \qquad 4.48$$

This relationship depends on the ratio of the two brightness levels.

4.6 IMPLEMENTATION

The artificial retina was implemented on an Acorn Archimedes computer using both the standard and the alternative photoreceptor formulations. Appendix 4 gives the details of the retina inputs and outputs, and the classification performance of the network.

4.7 DISCUSSION

The network has been tested on grey level inputs and apart from two cases, the images were fully classified. For the two cases where there was not full classification, the neuron output for the two correct neurons was far higher than for the other neurons. It could be argued that by adjusting the network retina parameters a better set of results could have been obtained.

The formulation of the retina used in these experiments was capable of performing brightness invariance filtering. On examination of the two photoreceptor relationships, the modified photoreceptor gave better results with respect to accuracy of classification and the contrast in the neuron output results. However, this formulation is likely to break down at very low light levels.

In a comparison of edge detection neural networks [Thiaville et al. 1990], the Mead retina was compared with a Canny-like model and a Deriche-like model. The results indicate that the Mead retina produced the poorest results and the Canny-like model out-performed the Deriche-like model. However, the Canny-like model uses *negative* resistors and the stability of the network must be considered. As well as edge detection the model was used to model intensity invariance.

4.8 CONCLUSIONS

The time-independent artificial retina equations can be modelled and solved using the above techniques.

Using both retina formulations the retina was able to classify the training set. When classifying the images with different brightness levels the retina with the alternative photoreceptor formulation had the better classification performance.

5.0 SCALE INVARIANT CLASSIFICATION

5.1 CHAPTER CONTENTS

In this chapter, the author built on the shift invariant network to develop a method for the classification of scale invariant objects that is also capable of determining the components of these objects. Using low level primitive detectors and having extra compensation in some neurons, the network could be taught without recourse to the hot spot that was used in Chapter 3.

A Neocognitron was also constructed for performance comparisons. Supervised learning was used to teach networks, therefore valid comparisons could only be made for classification performance.

5.2 SCALE INVARIANT MODEL

When developing the model for a scale invariant network, inspiration from biological systems was sought. The method by which the retina detected motion [Poggio and Christof 1987] (described in Chapter 2) was seen to be a simple and elegant solution to a complicated problem.

This mechanism uses direction sensitive neurons that have excitatory and inhibitory synapses in close spatial proximity. For the condition where only the excitatory synapse fires, there is an excitatory post-synaptic potential (EPSP) that will reach the cell body. However, if both synapses are activated, the inhibitory synapse will veto the effect of the excitatory synapse. The inhibitory mechanism is known as silent or shunt inhibition

because the effect of the inhibitory signal is only apparent when it is combined with an excitatory signal. This effect is summarised in Table 5.1.

Excitatory Signal	Inhibitory Signal	Output
Not active	Not active	Not active
Not active	Active	Not active
Active	Not active	Active
Active	Active	Not active

Table 5.1 Silent inhibition

Instead of perceiving the neuron just as a summing and firing device, the neuron can also have logical functionality. This property can be extended to help the neuron search for features.

This silent inhibition mechanism can be used to construct a neuron that is sensitive to the location of features in the previous layer. Considering a dendrite that extends radially from the axis of the neuron, at the location of each feature in the previous layer, the dendrite has two connecting synapses, excitatory and inhibitory. The excitatory synapse is located closer to the cell body than the inhibitory synapse. The excitatory pathways from the previous layer neurons have thicker (faster) dendrites than the inhibitory pathways.

When a feature neuron fires in the previous layer two signals are sent towards the synapses, the excitation signal will override the inhibitory signal and will travel along the dendrite towards the cell body. The excitatory signal will always override the effect of the inhibitory signal because it is positioned closer to the cell body and the signal arrives first at the synapse.

The dendrite will connect with a number of previous layer neurons and each time a previous layer neuron fires, an excitatory signal will travel along the dendrite towards the cell body. However, if a pulse travelling towards the cell body approaches synapses connecting with a previous layer neuron that has also fired, it will be prevented from travelling towards the cell body. This is because the excitatory synapse will have been activated and the slower inhibitory signal will block the dendrite on reaching the synapse. With this arrangement, signals from neurons closer to the cell body override signals that are further away. This is shown in Figure 5.1.

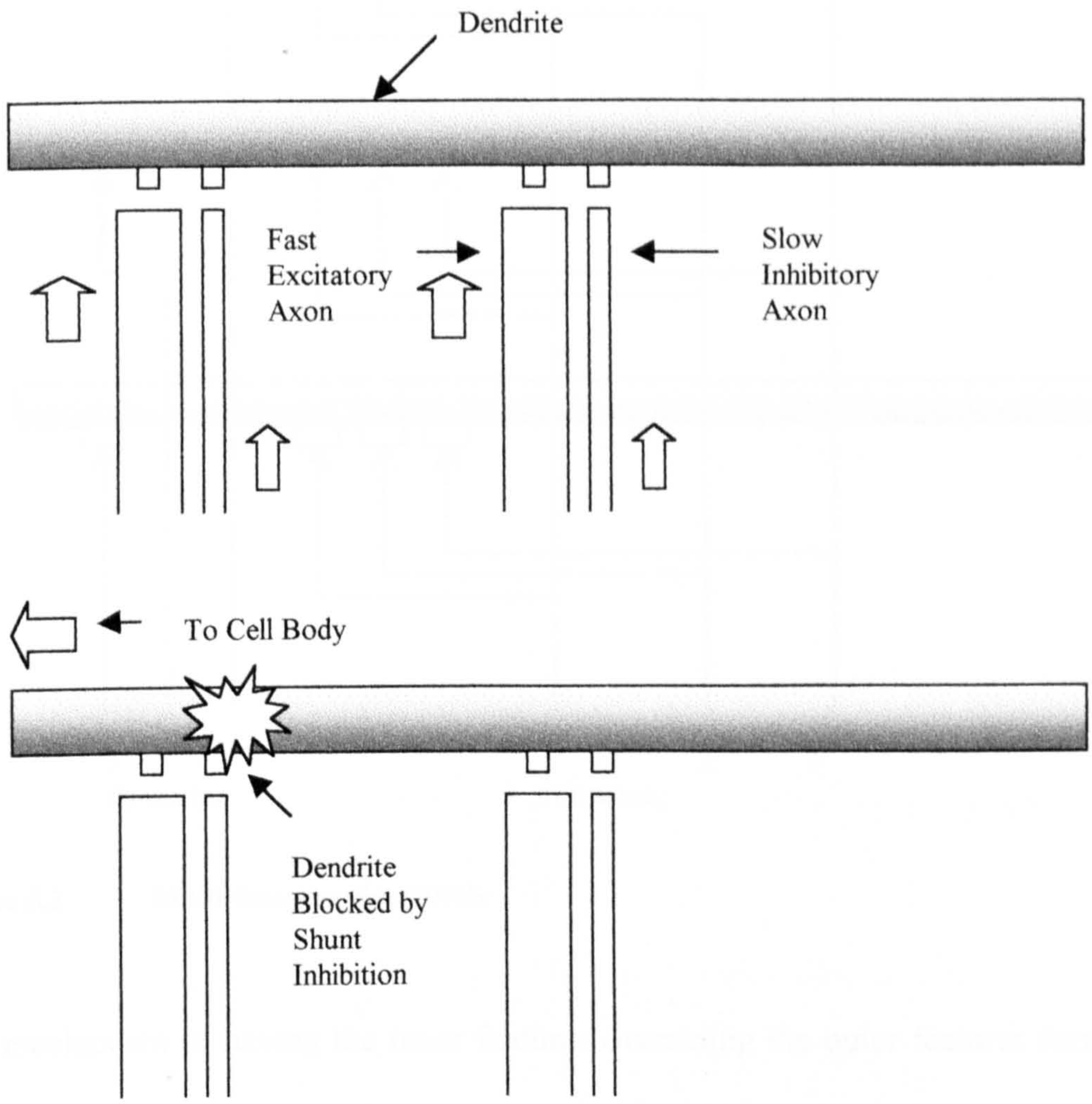


Figure 5.1 Overriding the outer neuron signals

The above system of overriding the signals from the outer neurons only considers one type of feature neuron in the previous layer. However, in practice, a feature detector will

receive inputs from a number of neurons. To accommodate these extra features the system has parallel dendritic trees, where each tree only has excitatory connections with one type of neuron. In contrast the inhibition system is fully connected so that there will be an inhibitory branch from each neuron to each dendrite. A strong value from any neuron will inhibit the outer neurons for every feature. This is shown in Figure 5.2.

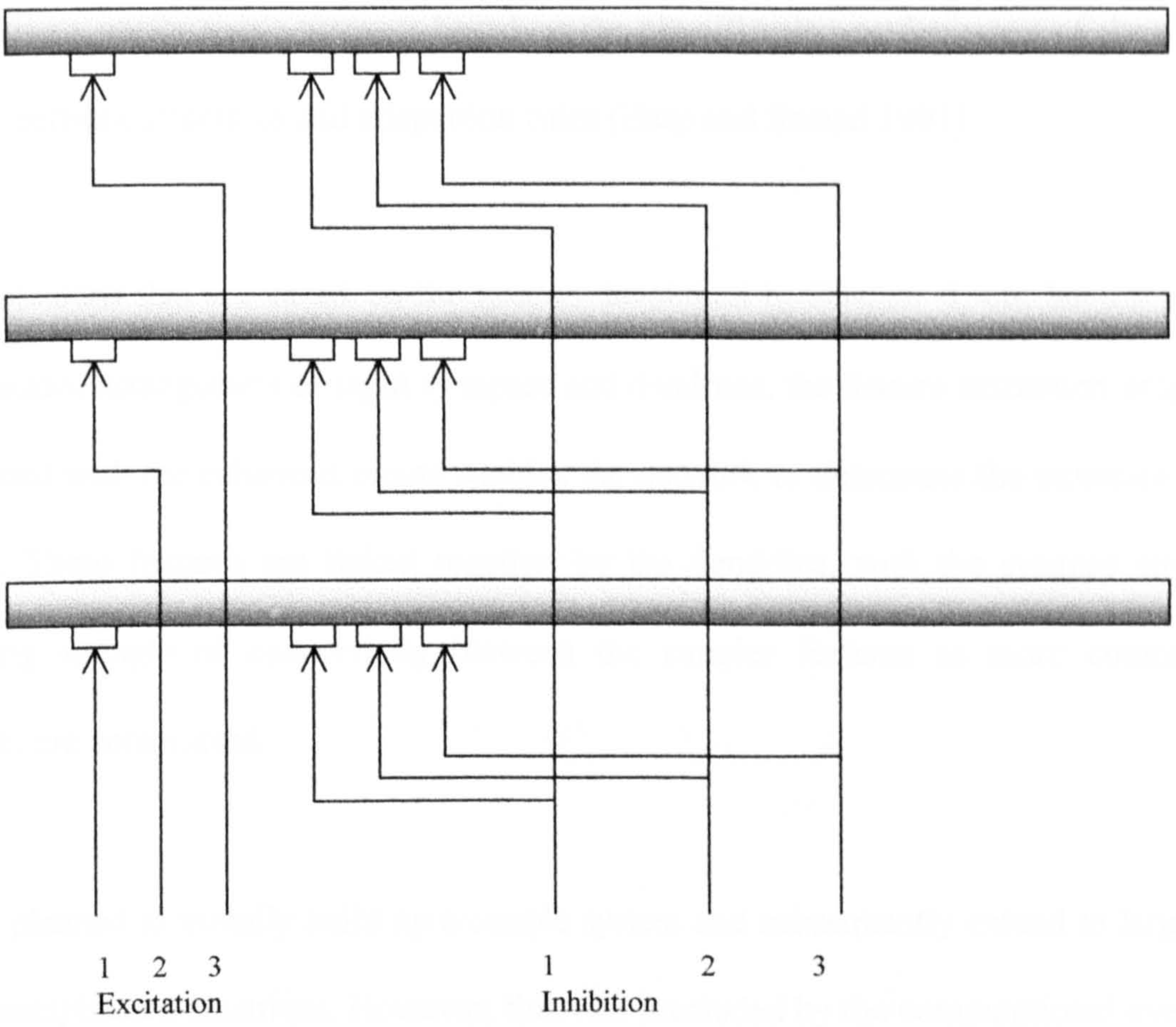


Figure 5.2 Multi-feature connectivity

The mechanism of having the inner features overriding the outer features forms the basis of the scale invariant network.

It was mentioned in Chapter 2 that the mammalian visual system is highly organised and is determined by the animal's genetic characteristics (genotype) which evolved over many generations. A similar approach is taken with the scale invariant network, where a network

is constructed from different types of neurons. However, instead of the network structure being genetically determined, reason and experimentation devised the structure. Adaptation (learning) rules were used to tune the network to facilitate the classification of scale invariant images.

It is possible to formulate the network structure and adaptation rules as a chromosome(s) and, using a performance measure based on the classification performance of the system, evolve network structures and adaptation rules [Harp and Samad 1991].

Using neurons that are based on the neuron developed in Chapter 4, but having a more complicated arrangement of input synapses and dendrites, the feature extraction properties combined with the enhanced inputs enables the network to determine the structure of the image. These features are linked together by the dendrites, with the synapse strengths denoting strength of connectivity between the simpler features as more complicated features are constructed.

It was planned to initially build up a simple system and subsequently extend to larger and more complicated situations. However, this was precluded by the computational overhead. Sometimes the network needed to be trained over many days. This would not present a problem in a biological system because of its inherent parallelism.

In view of heavy processing requirements, this analysis was restricted to small and simple systems. The restriction on the images to be processed are:

1. The images are constructed from thin lines.
2. The images are constructed from horizontal and vertical lines.

3. The images are noise free.

4. The images are constructed from binary pixel values.

5. Sub-features in the image are located at the mid-position of larger features.

The principle of operation of the network is:

As the signals pass from lower to higher layers, the information simplifies as it traverses a layer.

This simplification is caused by smaller features combining into a smaller number of larger features, and is illustrated in Figure 5.3.

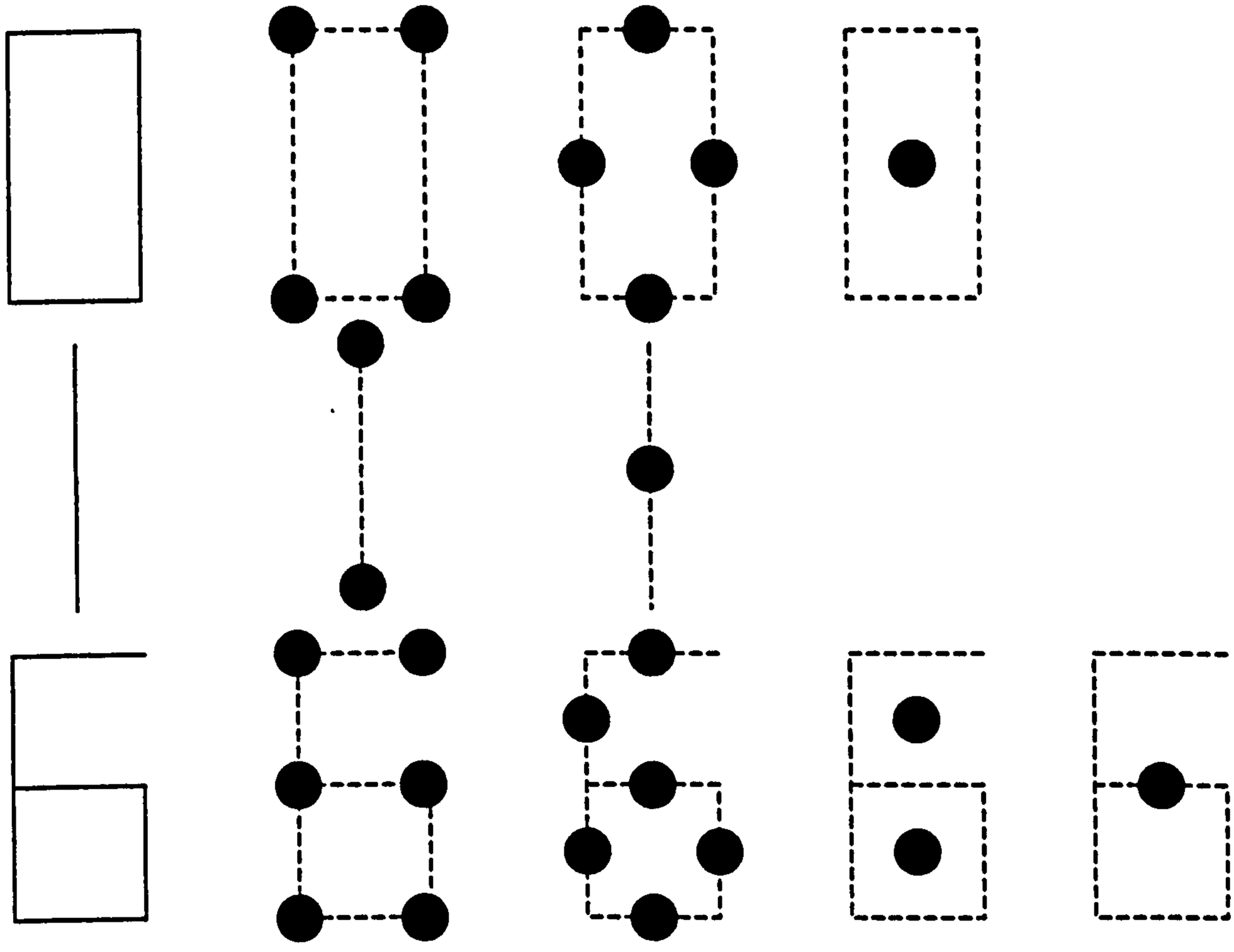


Figure 5.3 Examples of image simplification

Figure 5.3 shows simple line images for number 0, 1 and 6, and the processing stages.

The first stage is called the primitive layer, which identifies features in the images. Four corners were identified for the 0, two ends are identified for the 1, and for the number 6, one end, one tee and four corners are identified.

The next layer is called the size layer, the neurons locate the midpoints between certain features. Four lines were identified for the number 0, one for the 1 and six for the 6. At this point, the 1 has been classified.

The next layer is called the corner layer, where the neurons locate the corners connecting the centres of lines. One corner was identified for the 0 and two for the 6. At this point, the 0 has been classified.

Another size layer follows the corner layer. One line was identified to classify the number 6.

In the above description, three types of layer were used to classify the image: a primitive layer, two size layers and one corner layer. To classify images that are more complicated more layers are required. The general arrangement is a primitive layer followed by size, corner, size, corner and so on.

The complexity of the image can be defined as the number of non-primitive layers required to classify the image. Thus 0 has a complexity of two, 1 has a complexity of one and 6 has a complexity of three.

The activation of a layer for an image is the number of fully active non-coincident neurons in a layer. It is written as Al(n), where l denotes the layer and type (p, s1, c1, s2, or generic using x), and n denotes the image. For the image of number 0, the activation values are:

Ap(0) = 4

As1(0) = 4

Ac1(0) = 1

For the image of number 1

Ap(1) = 2

As1(1) = 1

For the image of number 6

Ap(6) = 6

As1(6) = 6

Ac1(6) = 2

As2(6) = 1

Using the above values as an illustration, a numerical criterion for the simplification of a network can be established.

$$A_{sx}(X) \leq A_{cx-1}, \; for \; x > 1 \qquad 5.1$$

$$A_{sl}(X) \leq A_{p}(X) \qquad 5.2$$

$$A_{cx}(X) < A_{sx}(X) \qquad 5.3$$

When training the position invariant network, the learning algorithm knew the location of the features of interest. This had a profound effect on the learning because it was easier to identify the neuron of interest and training was speeded up because only the outputs from a fraction of the neurons in a layer needed to be calculated and examined. For the size invariant network, a different approach was used in which the neurons have additional dendrites with fixed synapses. These synapses connect with specific neurons in the previous layers and tend to bias the output of the neurons to try to detect certain features. The fixed synapse dendrites look for patterns in the layers before the neuron's previous layer providing additional information to aid the learning process. These fixed synapse dendrites can be considered to be genetically determined in a similar manner to the primitives.

In operation, the neurons exhibit complex non-linear behaviour, with the neuron outputs being a mixture of the output from the fixed synapses and output from the untrained synapses. In a sea of neuron outputs, some outputs are biased in certain directions to aid training. This training further reinforces these differences in output and the neurons learn to identify specific features.

One way of considering this network is to consider that it is initially genetically organised, but it has the learning capacity to fine-tune and adapt itself to the environment.

In operation this network operates very slowly, because of the following reasons:

- The outputs from all of the neurons must be calculated when the network is training.

- There are additional fixed synapse dendrites that must be considered.

- The dendrites in the size and corner layers are branched and interact with many more neurons in the previous layer. This effectively reshapes the receptive fields of these neurons.

This extra neuron processing capacity significantly slowed the performance of the network. Modifications were made to the neurons to reduce the processing requirements. These modifications are listed below.

- Removal of the conductance parameter

  Removing the conductance parameter reduced the processing requirements and made it easier to understand the details of network training when examining the training log file.

- Removal of the size parameter

  Similar to removing the conductance parameter, removing the size parameter reduced the processing requirements and made it easier to understand the training.

- Simplification of the activation function

  The existing activation function used a square root, which is numerically intensive.

  In some layers, this activation function was replaced by an activation function that used the multiplication operator.

- Simplification of the distance criterion

  The neuron used a distance criterion that used square and square root functions. Two different types of criteria were examined in the hope of speeding up the processing in a neuron, these methods were named AND neuron and Fast Feature neuron.

Considering a neuron with one synapse, the existing neuron will have the input-output relationship that is shown in Table 5.2.

x	*w*	*Output*
0	0	1
0	1	0
1	0	0
1	1	1

Table 5.2 Feature neuron relationship

This is the NOT Exclusive-OR binary operator.

The AND neuron is based on the Boolean AND function that is shown in Table 5.3.

A	*B*	*Output* = *A* AND *B*
0	0	0
0	1	0
1	0	0
1	1	1

Table 5.3 AND Function

The AND function can be reformulated as

$$Output = \min(A, B) \qquad 5.4$$

Using this formulation the AND function can be applied to non-binary numbers, and was used in some neurons in the form:

$$Net' = \frac{\sum_{j=1}^{n} \min(x_j, w_j)}{\sum_{j=1}^{n} w_j} \qquad 5.5$$

The denominator provides the normalisation to ensure that *Net'* lies in the permissible range of 0 to 1.

The Fast Feature neuron is strongly based on the existing neuron, but uses a simpler form of discrimination. Instead of using the L2 Norm the L1 Norm is used, giving:

$$Net' = 1 - \frac{\sum_{j=1}^{n} |x_j - w_j|}{n} \qquad 5.6$$

Similarly, the denominator scales the values to give a permissible value for *Net'* of 0 to 1.0. For binary values the input-output relationship will be the same for the feature neuron shown in Table 5.2.

On examination of the two relationships, the fast feature neuron is symmetrical, that is the output depends on the error value, and is independent of the absolute values, whereas the AND neuron is biased to favour neurons with larger absolute values. This bias could be a useful to assist a neuron find a feature as it will respond better to the locations where the features are present.

Trials using the AND neuron for learning features indicated that the AND neuron had useful properties. The learning algorithms were still based on the feature neuron's mechanism, in which small value weights carry information. However these synapse values will have very little effect on the AND neuron classification, and its classification performance will be degraded.

Tests showed that when comparing the Fast Feature neuron with the AND neuron, the Fasts Feature neuron was better at classification but inferior at learning. At the end of a training run it was generally not able to correctly learn as many features as the AND neuron.

The weights in both systems behave in the same manner and are numerically equivalent. This property allowed the combined AND - Fast Feature neuron to be constructed. When learning the AND classifier was used, and when classifying the Fast Feature classifier was used. The detailed construction of the combined AND - Fast Feature neuron is used in the scale invariant layers and the details will be explained along with the layer configuration.

Three types of layer (primitive, size and corner) are used to construct the network. The alternation of size and corner neurons ends at the second size layer, called size2. These layers will be considered in greater detail.

5.2.1 PRIMITIVE DETECTOR LAYER

This is a layer of feature detectors that simply processes the information by classifying the inputs. The weights of these neurons are pre-programmed and no modification by the system is permitted. They are deemed to be similar to the line detectors discovered by [Hubel and Wiesel 1962]; although the primitives used here allow more complicated features than straight lines to be detected. These primitives can be assumed to be constructed solely by the influence of the animal's genetic code.

The receptive fields for all neurons are rectangular in shape and of constant dimensions. When detecting a line, only the synapses surrounding the line are important. As well as the superfluous synapses not being required for line detection, they can degrade performance by being affected by pixels that are not part of the line. To counteract this situation three synapse values were used;

Weight	Meaning
1	Accept 1 and reject 0
0	Accept 0 and reject 1
-1	Ignore the input

Table 5.4 Primitive weight values

Primitive detector synapses have a value of 1 where the line is expected to be, this line of 1s is surrounded by values of 0 and all other synapses have a value of -1. These detectors

are similar to the valley detectors used by [Pearson et al. 1986]. There are similarities with the simple cell receptive fields briefly described in Chapter 2.

For faster processing a fast activation function was used, which is expressed in terms of the value *Net* and the threshold τ.

For $0 \leq Net < \tau$

$$Output = \left(\frac{Net}{\tau}\right)^4 \qquad 5.7$$

For $\tau \leq Net \leq 1$

$$Output = Net \qquad 5.8$$

The neuron output calculated is shown in Appendix 5, Pseudocode A5.1.

5.2.2 LINE DETECTOR OR SIZE LAYER

The size layer uses neurons that are based on the input connectivity of the multi-feature connectivity and the combined AND - Fast Feature neuron. Similarly with the primitive layer, the features are arranged in groups at the same spatial location, with each neuron detecting a different feature. The dendrites extend from the neuron in two lines at 180° to each other.

Each neuron in a feature detector is given a number starting at 1, with the convention that the odd-numbered neurons have the dendrites arranged horizontally and even-numbered neurons have the dendrites arranged vertically. Examination of the feature detectors will give alternate detectors arranged at 90° to each other.

The arrangement of dendrites leaving a neuron and travelling in a straight line is called a group. Table 5.5 shows the group numbering convention used.

Neuron Search Direction	Group	Number Search Direction
Horizontal	1	Left
Horizontal	2	Right
Vertical	1	Down
Vertical	2	Up

Table 5.5 Search direction convention

The inputs and outputs to a size layer neuron are shown in Figure 5.4.

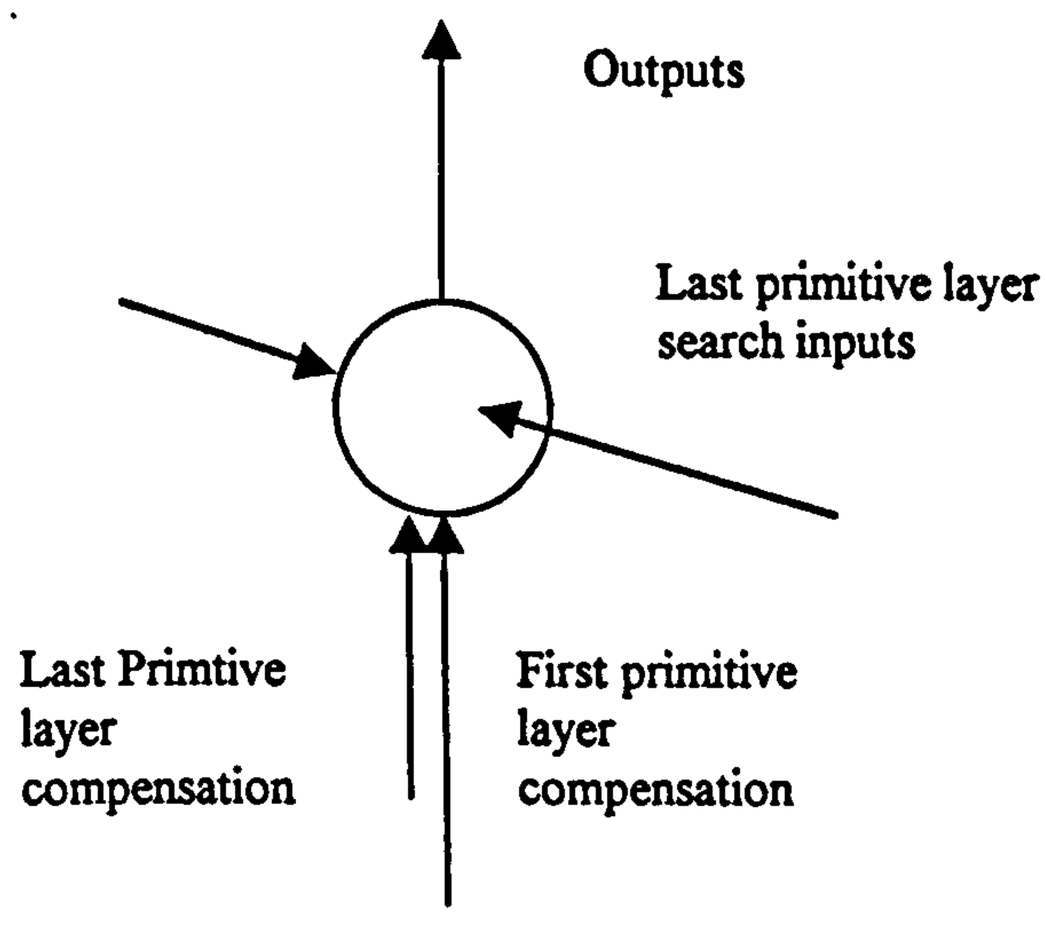


Figure 5.4 Size layer neuron inputs and outputs

When determining the output from a neuron the first stage is to determine the inputs to the cell body. The algorithm for searching the size layer dendrites is shown in Appendix 5, pseudocode A5.2.

The neuron outputs are calculated using the algorithm shown in Appendix 5, Pseudocode A5.3. The activation function is the same as the activation function for the primitive layer neurons.

Three types of compensation were used to bias the neuron to give a larger output when it encounters favourable inputs:

- Primitive mid-point compensation

  This compensation is used to favour neurons that are located at a mid-point of a previous layer feature. This compensation assists in the identification of the correct features that are selected for training, hence only the AND neuron is compensated in this manner.

- Contribution from retina

  This compensation reads the values from the retina. The position of the neuron is transformed to the position on the retina and the output from a line of photoreceptors is summed and divided by the maximum possible output. This line consists of three photoreceptors giving a maximum value of three.

- Contribution from previous layer

  This compensation receives a value from the last primitive layer. At the location of the size neuron, the processed outputs from all of the neurons in the feature

detector are read and the maximum value is determined. This value is compared with 0.85 and if found to be greater, the value 1.0 is returned otherwise 0.0 is returned. This type of contribution is sensitive to the presence of a feature.

When the contribution is subtracted from 1.0 and multiplied with a value, the product has a value either of its original value or 0.0. This type of compensation tends to suppress neurons that are coincident with neurons that are highly active in the last primitive layer.

5.2.3 CORNER LAYER

The corner layer is similar to the size layer in that it uses neurons that are based on the input connectivity of the multi-feature connectivity and the combined AND - Fast Feature neuron. However, the dendrites extend from the neuron in two lines at 90° to each other.

Similarly with the size layer, neurons in the corner layer search in different directions depending on the neuron number. However instead of two search directions, there are four search directions known as left bottom, right bottom, right top and left top;

- Left Bottom

  The neuron searches for a bottom left hand corner by having one dendrite branch searching downwards and the other dendrite branch searching to the left.

- Right Bottom

  The neuron searches for a bottom right hand corner by having one dendrite branch searching downwards and the other dendrite branch searching to the right.

- Right Top

  The neuron searches for a top right hand corner by having one dendrite branch searching upwards and the other dendrite searching to the right.

- Left Top

  The neuron searches for a top left hand corner by having one dendrite branch searching upwards and the other dendrite searching to the left.

Each neuron in a feature detector is given a number starting at 1. The convention used is that the neurons numbered 1, 5, 9, ... search for the left bottom features, the neurons numbered 2, 6, 10, ... search for the bottom right features, the neurons numbered 3, 7, 11, ... search for the top right features, and the neurons numbered 4, 8, 12, ... search for the top left features.

The inputs and outputs for a corner layer neuron are shown in Figure 5.5.

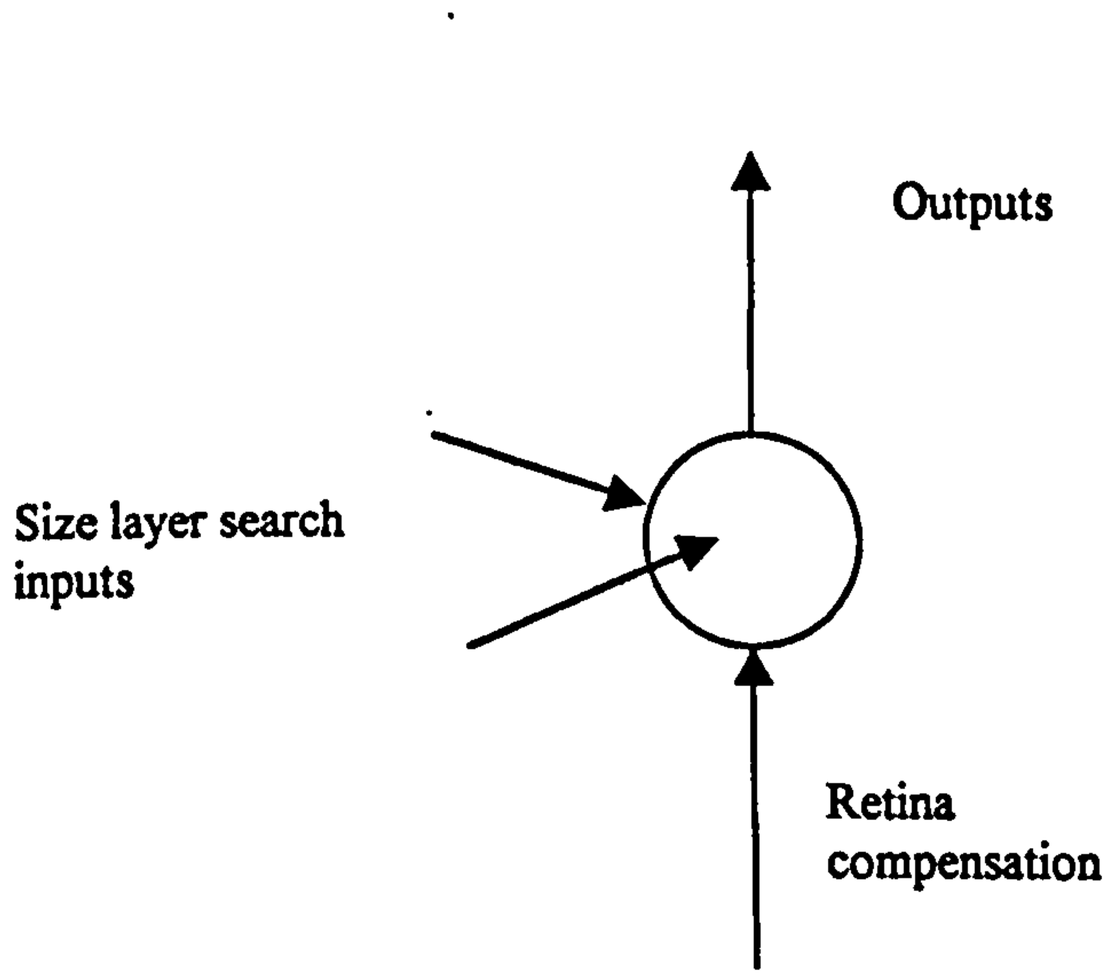


Figure 5.5 Corner layer neuron inputs and outputs

When determining the output from a neuron the first stage is to determine the inputs to the cell body. The algorithm used to search the dendrites is shown in Appendix 5, Pseudocode A5.4.

The activation function is the same as the activation function for the primitive layer neurons.

The neuron outputs are calculated using the algorithm shown in Appendix 5, Pseudocode A5.5.

Compensation by reading the values in the retina was used to bias the neuron to give a larger output when it encounters favourable inputs. This compensation returned a value that depended on the type of feature in the corner layer. The general methodology is to sum the outputs from two lines and divide by the maximum possible output. In the experiments, a line length of three pixels was used, and the two lines give a maximum of six pixels. Hence, the returned value is the total output divided by six, where a value of 1.0 indicates that there were lines coincident with the compensation detectors and a value of 0.0 indicates that the retina values at the compensation detectors were zero.

The location of the compensation detectors is at the position of the maximum outputs from the size layer. For each type of feature, the feature detector input pattern is as follows:

- LEFT BOTTOM

  At the position of left seeking dendrite, a vertical line is detected and at the position of the bottom seeking dendrite, a horizontal line is detected.

- RIGHT BOTTOM

  At the position of right seeking dendrite, a vertical line is detected and at the position of the bottom seeking dendrite, a horizontal line is detected.

- RIGHT TOP

  At the position of right seeking dendrite, a vertical line is detected and at the position of the top seeking dendrite, a horizontal line is detected.

- LEFT TOP

  At the position of left seeking dendrite, a vertical line is detected and at the position of the top seeking dendrite, a horizontal line is detected.

This compensation favours neurons that have dendrite branches that are perpendicular to lines on the retina.

The activation function is based on the activation function as specified in Chapter 3.

5.2.4 SIZE 2 LAYER

The size 2 layer uses neurons that are based on the input connectivity of the multi-feature connectivity and the combined AND - Fast Feature neuron. Similarly with the size layer, there are two dendrites that extend from the neuron body at 180° to each other.

The inputs and outputs for a size 2 layer neuron are shown in Figure 5.6.

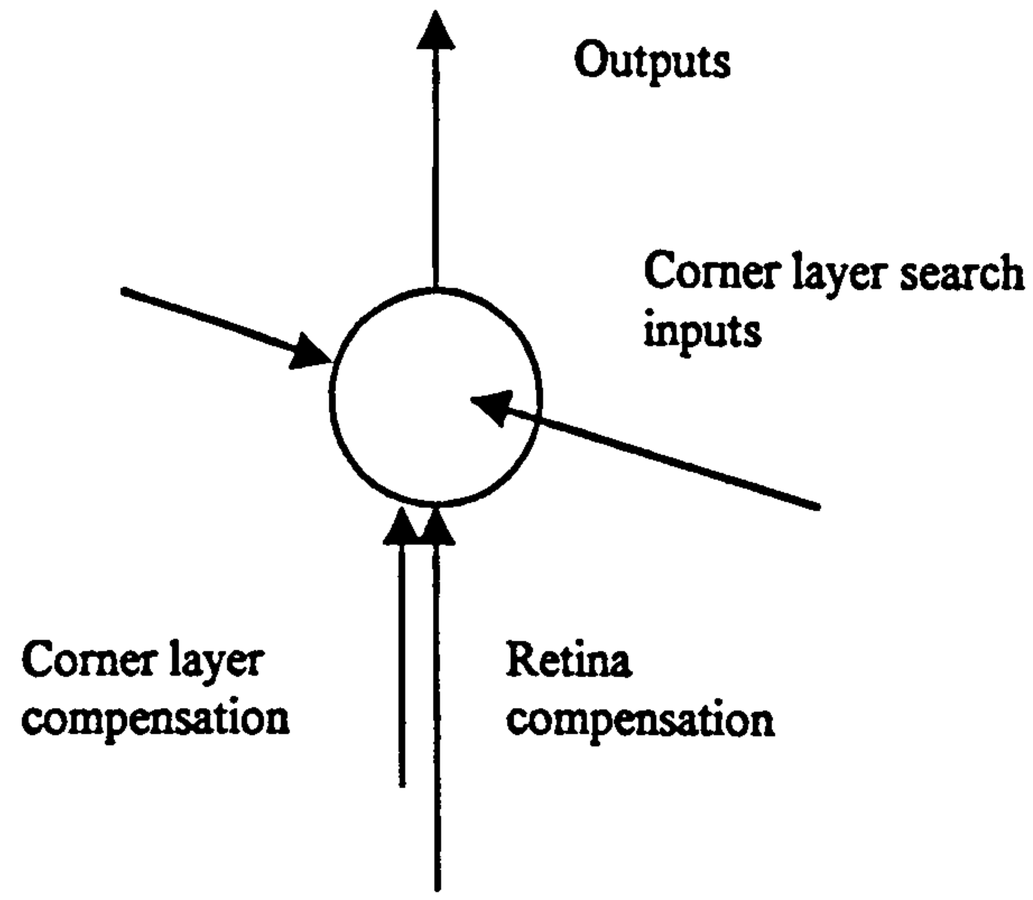


Figure 5.6 Size 2 layer neuron inputs and outputs

When determining the output from a neuron the first stage is to determine the inputs to the cell body. The algorithm for searching the size 2 layer dendrites is shown in Appendix 5, pseudocode A5.6.

The activation function is the same as the activation function for the corner layer neurons. The neuron outputs are calculated using the algorithm shown in Appendix 5, Pseudocode A5.7.

Three types of compensation were used to bias the neuron to give a larger output when it encounters favourable inputs:

- Corner layer mid-point compensation

  This compensation is used to favour neurons that are located at a mid-point of two corner layer features that are close together. This compensation assists in the identification of the correct features that are selected for training, hence only the AND neuron is compensated in this manner.

- Contribution from retina

  This compensation reads the values from the retina. The position of the neuron is transformed to the position on the retina and the output from a line of photoreceptors is summed and divided by the maximum possible output. This line consists of three photoreceptors giving a maximum value of three. The search direction along the retina is at right angles to the direction of search on the corner layer.

5.3 STRUCTURAL DECOMPOSITION

Structural decomposition involves producing a description of all of the features located by the classification process. The neurons in the network contain the feature information and, to produce the description of the features, descriptions for the salient features are read from a look-up table. This look-up table is manually populated and simply involves assigning a description to each feature number in each layer. Apart from any hard-wired primitives, the features can only be assigned after training. This is because the network is taught by unsupervised learning and which neurons learn which features cannot be known in advance.

This is in contrast to how the features are named in MIRABELLE [Mohr 1990].

5.4 SIZE INVARIANT EXPERIMENTS

Appendix 6 demonstrates the learning capabilities of the size invariant network and, using the learnt synapse values and determined network features, the network attempts to classify the original training set and a similar set of images with a slightly increased size.

A summary of the final layer results for the 18 × 18 retina is shown in Table 5.6.

Image Number	Image	Maximum Output
1	Number 0	1.000
2	Number 1	0.808
3	Number 2	1.000
4	Number 3	0.634
5	Number 4	1.000
6	Number 5	1.000
7	Number 6	1.000
8	Number 7	1.000
9	Number 8	1.000
10	Number 9	1.000

Table 5.6 Summary of the final layer results for 18 × 18 retina

For comparison purposes, the author constructed a Neocognitron that learns by unsupervised learning. The Neocognitron was as described in [Fukushima et al. 1983], with the fixed weights obeying the relationship shown in Equation 5.7.

$$w \propto f^{r} \qquad 5.7$$

Where,

f is the weight fall-off

r is the distance from neuron axis

The Neocognitron was tested against the same images as before. Details of the network's construction, learning parameters and classification performance are given in Appendix 6.

5.5 DISCUSSION

This discussion firstly considers the results of the training set learning and classification followed by a discussion of classification of the larger images.

5.5.1 TRAINING SET LEARNING AND CLASSIFICATION

Examination of the results of the primitive layer showed that the size of the retina was too small for the images. This was demonstrated by the classification of the number 4, in which the lowest vertical line was not completely classified. The size of the retina was restricted because of the long network training time. Training the size layer took about 24 hours, and the training of the whole network was carried out over three days. Some tests were carried out using a larger retina and the total training time took about five days.

The size layer was able to train 23 out of the 24 available feature neurons. On examination of the results, it was discovered that all the images were classified in a predicted manner except for the following images:

- Number 1

  Bottom right foot

- Number 4

  To the right and bottom of the cross

- Number 5

  Top horizontal bar

- Number 6

  Top horizontal bar

- Number 8

  Middle horizontal bar

Looking at the corner layers, 14 out of the 16 feature neurons were taught a definite feature. However out of these 14 features, three feature neurons learnt features that express low outputs and have correspondingly low synapse values.

The output from the corner layer was as predicted except for the following features:

- Number 1

  Bottom left hand corner was not fully classified

- Number 2

  There were three fully classified regions. The extra feature was situated on the middle horizontal bar.

- Number 4

  Features around the cross are not classified

- Image 6

  The fully classified features did not align vertically

- Number 8

  A feature was classified that was not readily identified with a lower level feature and the expected upper rectangle was not fully classified.

The images for numbers 0 and 7 were identified in this layer.

When considering the size 2 layer the classification of images 1 and 8 are ignored because they have been classified in the corner layer. The classification behaved as expected apart from the following number images:

- Number 1

  The image was not completely classified

- Number 4

  No image was readily identified

The classification of the image 5 has always been a problem, this was probably because it was more complicated than the other images and the retina was too small for complete identification. This image was the only image to have a cross, leading to a low occurrence of this feature during the training session and a lower probability of this feature being classified.

The low output for image 2 was rather disappointing and indicates that there were difficulties learning this feature.

5.5.2 LARGER IMAGES CLASSIFICATION

This discussion is for the larger images on a 20 × 20 retina. As expected the performance of the primitive layer was similar to that of the training set.

A summary of the final layer results for the 20 × 20 retina is shown in Table 5.7.

Image Number	Image	Maximum Output
1	Number 0	1.000
2	Number 1	0.795
3	Number 2	1.000
4	Number 3	1.000
5	Number 4	0.340
6	Number 5	1.000
7	Number 6	1.000
8	Number 7	1.000
9	Number 8	0.621
10	Number 9	1.000

Table 5.7 Summary of the final layer results for 20 × 20 retina

Examination of the size layer reveals that the classification performance is very similar to that of the training set. Indeed the same features are not fully classified. This strongly suggests that the size layer has scale invariance properties.

Examination of the corner layer reveals that it has similar classification performance as for the training set.

- Number 2

  Slight deterioration of the output levels with the additional bar still present

- Number 4

  Slight deterioration in output levels

- Number 6

  Very slight deterioration in output levels

- Number 8

  Slight deterioration in output levels with a different distribution of high output neurons

- Number 9

  Only one neuron is fully classified

Examination of the size 2 layer reveals the classification performance is very similar as for the training set except for the classification of image 9. The poor performance of classifying image 9 was due to the different distribution of high level output neurons in the corner layer. Further work needs to be carried out to improve the performance when classifying the number 8.

The results shown in Appendix 6 demonstrated the following principles:

- Unsupervised learning can be used to train three layers of a neural network
- The network represents a hierarchical structure
- Neurons in each layer can represent a distinct feature
- The image classified by the network can be automatically decomposed into simpler features

The system was not able to classify the number 4, the enlarged number 8 and noisy images.

Comparison tests using a Neocognitron were also carried with the results shown in Table A5.4. The low output level was due to the form of the complex layer equations. For a fair comparison, it should be imagined that they are multiplied by 2. As expected the training set was properly classified, but the system started to break down with the larger images. It misclassified the larger number 4 and the output for the number 0 was significantly down.

Using unsupervised learning made the network training unduly difficult. A lot of time was spent trying to formulate suitable learning rules. This was contrasted with supervised training of the Neocognitron. It was discovered that the training images for each layer had to be correct, otherwise the higher layers would not be taught properly. Once the behaviour of the network was known, training the network became relatively straightforward. Indeed this training was rather contrived, in that the training set was formulated in a way that would place the features in the correct position for hierarchical classification. On this basis, it was not a fair comparison but it allowed classification performance to be assessed.

Examination of the output from the lower layers revealed that the first layer gave sharp boundaries where the neurons had located features. However, the higher the level, the more spread out the output levels became, the maximum output at each neuron location being approximately the same, with the main difference being the plane associated with the maximum output neuron. Constructing a structural system based on these layer output levels would be very difficult. This spreading mechanism gave the Neocognitron the ability to classify the larger images. This mechanism will eventually break down with larger images.

In contrast, the size invariant network developed here gave good demarcation between classifying and not classifying neurons. The size invariant network was also able to give the locations of the features, while this information was lost in the Neocognitron. On the other hand, the Neocognitron is more robust.

The two systems appear to have complementary levels of performance.

6.0 DISCUSSION, CONCLUSIONS AND RECOMMENDATIONS

This research into image classification invariance addressed three problems: translation invariance, brightness invariance and scale invariance. The investigations into translation and brightness invariance were used to test the learning algorithms and the grey scale performance of the network. The translation invariance mechanism was used to identify similar features at different locations in the scale invariant network. Although the brightness invariance mechanism was not used in the final network architecture, it is a valid test in its own right.

6.1 TRANSLATION INVARIANCE

The translation invariance was tested out on two network topologies: a single layer and a triple layer network. Although unsupervised learning was used, there was still a measure of supervision in that the location of the object in the image was given as part of the teaching schedule. Both clean and noisy images were tested with the performance of the single layer network having the better performance.

This network used a neuron that was based on the error criteria used in template matching and was capable of feature classification in one layer.

The author, as part of this research, has developed novel unsupervised learning techniques, namely:

- Tiring the neuron after learning
- Splitting the learning mechanism into three types: updating, moving and clustering.

This type of classifier was used to form the basis of the scale invariance network.

6.2 BRIGHTNESS INVARIANCE

When testing the brightness invariance, a translation invariance network was used with an artificial retina based on the silicon retina in [Mead and Mahowald 1988 and Mahowald and Mead 1991]. The retina was simulated using a time invariant mathematical model that was solved using an iterative process. By using a suitable logarithmic transfer function, it was possible to classify the images in a brightness invariant manner.

Good grey scale performance is required for the scale invariant layer.

6.3 SCALE INVARIANCE

The scale invariant network used a radically different network that combined the elements of structural techniques and neural networks. By adding suitable compensation elements to the network, it was possible to have fully unsupervised training, in which the network was able to determine the location of the salient features.

Perhaps the most remarkable aspect of the scale invariant network was the unsupervised learning mechanism. By giving the network a few *hints*, it was able to learn most of the structure of the images.

Two main aspects of the network limit the effectiveness of the network to the laboratory, namely only being able to process simple images made from horizontal and vertical lines, and its poor robustness to noise. These characteristics of the system were due to the processing speed of the computer that allowed a small number of neurons, possessing a

rather coarse representation with very little redundancy in the system. Using a computer with an order of magnitude faster processing speed would allow more search directions and by searching a wedge shape of input signals, a tolerance could be built into the system.

Because of the heavy processing requirements and the complexity of the network, the image set was restricted to small simple images that were constructed from horizontal and vertical lines.

It was discovered that the network was able to learn and classify the numbers 0, 1, 2, 3, 5, 6, 7, 8, and 9. The network was not able to classify the number 4. When testing with enlarged images, the performance was similar to classifying the training set except for the classification of number 8.

In addition to classifying the images, the system was also able to produce a structural decomposition of the images.

The Neocognitron correctly classified eight out of the ten larger images, and partially classified another. This performance was at the expense of the feature location information.

These two networks behaved in a complementary manner. Whereas the size invariant network was built around having a structural representation of the image, the Neocognitron was based around tolerance to the input features. It would not be unreasonable to incorporate the tolerance elements of the Neocognitron into the size invariant network, perhaps with good effect.

6.4 CONCLUSIONS

The translation invariant network was able to demonstrate the following principles:

- Unsupervised learning can be used to train a translation invariant network
- The network can classify images invariant of the translation

The brightness invariant system using an artificial retina was able to demonstrate the following principles:

- The artificial retina was able to provide a brightness invariant pre-processing layer
- The Lanczos vectors method used to solve the matrix equations gave good results

The scale invariant network was able to demonstrate the following principles:

- Unsupervised learning can be used to train three layers of a neural network
- The network was able to classify images in a scale invariant manner
- The network represents a hierarchical structure
- Neurons in each layer can represent a distinct feature
- The image classified by the network can be automatically decomposed into simpler features

6.5 RECOMMENDATIONS FOR FURTHER WORK

Although the scale invariant network was able to classify images in a scale invariant manner and perform structural decomposition, the network had problems that made it unsuitable for commercial applications. With the availability of more powerful computers, it will be easier to make improvements to the network.

Further research in this area might wish to concentrate on the following areas:

- Classification robustness

  The classification robustness was not very good. This was probably because the dendrites that searched for previous layer features only searched in either horizontal or vertical directions. By utilising a finer network mesh and using search methodology that had a wider search pattern than just one neuron, the network should have a measure of tolerance and be more robust.

- Automatic network topology

  The network topology was determined by human intervention and some of the network features, especially the neuron compensation, were developed by an iterative method. It is considered that the topology could be automatically determined using an evolutionary approach such as genetic algorithms. This would lend validity to the argument used, when developing the network topology, that the network architecture was genetically determined and the learning process fine tunes the system. Genetic algorithms have been used to design neural network architectures [Harp and Samad 1991], and this approach could be extended to the scale invariant network.

- Unsupervised learning

  Although the unsupervised learning mechanisms performed well, improvements could be made to provide better mapping between the features and the neurons.

REFERENCES

Alkon, D. L.

"Memory Storage and Neural Systems", Scientific American, July 1989, pp 26-34.

Al-Shaykh, O. K., and Doherty, J. F.

"Invariant Image Analysis Based on Radon Transform and SVD", Feb. 1996, IEEE Transactions on Circuits and Systems-II: Analog and Digital Signal Processing, Vol. 43, No. 2, pp 123 – 133.

Aleksander, I., and Stonham, T. J.

"Guide to pattern recognition using random-access memories", Computer and Digital Techniques, February 1979, Vol. 2, No. 1, pp 29 - 40.

Andre, D.

"Learning and Upgrading Rules for an OCR System Using Genetic Programming", 1994, Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE Press, Vol. 1, pp 462-467.

Austin, J., and Stonham, T. J.

"Distributed associative memory for use in scene analysis", Image and Vision Computing, Vol. 5. No. 4, 1987, pp 251-260.

Barnard, E., and Casasent, D.

"Shift Invariance and the Neocognitron", Neural Networks 1990, 3(4), pp 403 – 410.

Boyle, R. D., and Thomas, R. C.

"Computer Vision A First Course", Blackwell Scientific Publications, 1988, ISBN 0-632-01577-2.

Broomhead, D. S., and Lowe, D.

"Multivariable Functional Interpolation and Adaptive Networks", 1988, Complex Systems, 2 pp 321 – 355.

Bruce, V., and Green, P. R.

"Visual perception: physiology, psychology and ecology. – 2nd ed.", 1990, Pub. Lawrence Erlbaum Associates Ltd., ISBN 0-86377-146-7, pp 128 –131.

Bunke, H. (a)

"String Grammars for Syntactic Pattern Recognition", Syntactic and Structural Pattern Recognition Theory and Application, 1990, ISBN 9971-50-566-5, pp 29 - 54.

Bunke, H. (b)

"String Matching for Structural Pattern Recognition", Syntactic and Structural Pattern Recognition Theory and Application, 1990, ISBN 9971-50-566-5, pp 119 - 144.

Carpenter, G. A. (a), and Grossberg, S.

"A Massively Parallel Architecture for a Self-Organising Neural Pattern Recognition Machine", Computer Vision, Graphics, and Image Processing, 37, 1987, pp 54 - 115.

Carpenter, G. A. (b), and Grossberg, S.

"ART2: self-organization of stable category recognition codes for analog input patterns", Applied Optics, Vol. 26, No. 23, December 1987, pp 4919 - 4930.

Carpenter, G. A. (c), and Grossberg, S.

"Invariant Pattern Recognition and Recall by an Attentive Self-Organising ART Architecture in a Nonstationary World", Proceedings 1st IEEE Conference on Neural Networks, Vol. 2, 1987, pp II-737 - II-745.

Carpenter, G. A., and Grossberg, S.

"The ART of Adaptive Pattern Recognition by a Self-Organizing Neural Network", Computer, March 1988, pp 77 - 88.

Chen, S., Cowan, S. F. N., and Grant, P. M.

"Orthogonal least squares learning algorithm for radial basis functions network", IEEE Transactions on Neural Networks, 2, 302 - 309, 1991.

DeSieno, D.

"Adding A Conscience To Competitive Learning", Proceedings of the IEEE International Conference on Neural Networks, 1988, pp I-117 - I-124.

Dessimoz, J.-D., Birk, J. H., Kelley, R. B., Martins, H. A. S., and I, C. L.

"Matched Filters for Bin Picking", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-6, No. 6, November 1984, pp 686-697.

Eeckman, F. H., Colvin, M. E., and Axelrod, T. S.

"A Retina-Like Model for Motion Detection", IEEE INNS International Conference on Neural Networks, 1989, Vol. 2, pp II-247 - II-249.

Elliman, D. G., and Banks, R. N.

"Shift invariant neural net for machine vision", IEE Proceedings, Vol. 137, Pt. 1, No. 3, June 1990, pp 183 - 187.

Fischbach, G. D.

"Mind and Brain", Scientific American, September 1992, Vol. 267, No. 3, p24-33.

Fukumi, M., Omatu, S., Takeda, F., and Kosaka, T.

"Rotation-Invariant Neural Pattern Recognition System with Application to Coin Recognition", IEEE Transactions on Neural Networks, March 1992, Vol. 3, No. 2, pp 272 - 279.

Fukushima, K., Miyake, S., and Ito, T.

"Neocognitron: A Neural Network Model for a Mechanism of Visual Pattern Recognition", IEEE Transactions on Systems, Man, and Cybernetics, September/October 1983, Vol. SMC-13, No. 5, pp 826 - 834.

Fukushima, K.

"A Neural Network for Visual Pattern Recognition", Computer, March 1988, pp 65 - 75.

Fukushima, K., and Imagawa, T.

"Recognition and Segmentation of Connected Characters With Selected Attention", Neural Networks, 1993, Vol. 6, pp 33 - 41.

Gonzaga, A., and Costa, J. A. F.

"Moment invariants applied to the recognition of objects using neural networks", 1996, SPIE Vol. 2847, pp 223 – 233.

Govindan, V. K., and Shivaprasad, A. P.

"Character Recognition - A Review", Pattern Recognition, 1990, Vol. 23, No. 7, pp 671-683.

Grimes, C. A.

"Using Genetic Techniques in the Planning of Railway Track Maintenance Work", Genetic Algorithms in Engineering Systems: Innovations and Applications, 13-14 September 1995, pp 293-298.

Grimes, C. A., Picton P. D., and Elliman D. G.

"A neural network position independent multiple pattern classifier", Artificial Intelligence in Engineering, 1996, Vol. 10, pp 117-126.

Harp, S. A., and Samad, T

"Genetic Synthesis of Neural Network Architecture", Handbook of Genetic Algorithms, by Lawrence Davis, Van Nostram Reinhold, 1991, ISBN 0-442-00173-8, pp 202-221.

Harvey, P.

"Computer Science", Norman Price Publishers Ltd., 1971, ISBN 0 85380 013 8.

Hebb, D. O.

"The Organization of Behaviour: A Neuropsychological Theory", John Wiley, 1949.

Hestenes M. R., and Stiefel E.

"Methods of conjugate gradients for solving linear systems", J. Res. Nat. Bur. Standards, 1952, Vol. 49, pp 409 – 436.

Hoffman, J., Skrzypek, J, and Vidal, J. J.

"Cluster Network for Recognition of Handwritten, Cursive Characters", Neural Networks, Vol. 6, pp 69 – 78, 1993.

Hrycej, T.

"Unsupervised learning by backward inhibition", 11th Joint International Conference on Artificial Intelligence, 1989, pp 170 - 175.

Hu, M. K.

"Visual pattern recognition by moment invariants", IEEE Transactions on Information Theory, 1962, 8, 179 – 187.

Hubel, D. H., and Wiesel, T. N.

"Receptive fields and functional architecture of monkey striate cortex", Journal of Physiology, 1968, 195, pp 215 – 243.

Hutton, P. J.

"Handwriting recognition: a gentle introduction", The Computer Bulletin, December 1989, pp 18-21.

Kandel, E. R. and Hawkins, R. D.

"The Biological Basis of Learning and Individuality", Scientific American, September 1992, Vol. 267, No. 3, pp 52 - 60.

Kangas, J. A, Kohonen, T. K., and Laakson, J. T.

"Variants of Self-Organising Maps", IEEE Transactions on Neural Networks, Vol. 1, No. 1, March 1990, pp 93 - 99.

Kawashima, S., and Ishikawa, M.

"Locally Shift Invariant and Globally Location Dependent Recognition of Complex Figures by Neural Networks", Methodologies for the Conception, Design, and Application of Intelligent Systems, Proceedings of IIZUKA 96, 1996, pp 478 – 481.

Kohonen, T.

"Self-organising and associative memory", Series in Information Sciences, Vol. 8, Spinger Verlag, 1984.

Kohonen, T.

"The "Neural" Phonetic Typewriter", Computer, March 1988, pp 11 - 22.

Koza, J. R.

"Genetic Programming: on the programming of computers by means of natural selection", 1992, ISBN 0-262-11170-5.

Kröner, S.

"Adaptive averaging in higher order neural networks for invariant pattern recognition", IEEE International Conference on Neural Networks ICNN 95, Nov – Dec 1995, pp 2438 – 2443.

Lanczos C.

"An iteration method for the solution of the eigenvalue problem of linear differential and integral operators", J. Res. Nat. Bur. Standards, 1950, Vol. 49, pp 255 – 282.

Lee, B., Cho, Y., and Cho, S.

"New invariant pattern recognition system based on pre-processing and reduced second-order neural network", IEEE International Conference on Neural Networks ICNN 95, Dec 1995, pp 2099 – 2103.

Lee, S-K., and Jang, D.

"Translation, Rotation and Scale Invariant Pattern Recognition Using Spectral Analysis and Hybrid Genetic-Neural-Fuzzy Networks", 1996, Computers and Industrial Engineering, Vol. 30, No. 3, pp 511 – 522.

Lenz, R.

"Group Invariant Pattern Recognition", Pattern Recognition, Vol. 23. No. 1/2, 1990, pp 199 – 217.

Li, W., and Nasrabadi, N. M.

"Invariant Pattern Recognition Based on a Neural Network of Cascaded RCE Nets", IEEE INNS International Joint Conference on Neural Networks, 1990, Vol. 2, pp II 845 - II 853

Liatsis, P., and Goulermas, Y. J. P.

"Minimal Optimal Topologies for Invariant Higher-Order Neural Architectures Using Genetic Algorithms", IEEE International Symposium on Industrial Electronics ISIE 95, July 1995, pp 792 – 797.

Lin, W.-G., and Wang, S.-S.

"A New Neural Model for Invariant Pattern Recognition", Neural Networks, Vol. 9, No. 5, 1996, pp 899 – 913.

Lippmann, R. P.

"An Introduction to Computing with Neural Nets", IEEE ASSP Magazine, April 1987, pp 4 – 22.

MacQueen, J.

"Some methods for classification and analysis of multivariate observations", Proceedings Fifth Berkeley Symposium on Math. Stat. and Prob. 1, pp 281 - 287, 1967.

Mahowald M. A., and Mead C.

"The Silicon Retina", Scientific American, May 1991, Vol. 264, No. 5, pp 40 – 46.

Marr, D.

"Vision", W. H. Freeman and Company, 1982, ISBN 0-7167-1567-8.

Maunsell, J. H. R., and Newsome, W. T.

"Visual processing in monkey extrastriate cortex", Annual Review of Neuroscience, 1987, 10, pp 363 – 401.

McCulloch, W. S., and Pitts. W.

"A Logical Calculus of the Ideas Imminent in Nervous Activity", Bulletin of Mathematical Biophysics 5, 1943, pp 115 - 133.

McNeil, A. R., and Sarkodie-Gyan, T.

"A Performance Analysis and Optimisation of Shape Classification using an Edge Detected Contour Sequence and an Artificial Neural Network", 28th International Symposium on Automotive Technology and Automation Proceedings for the Dedicated Conference on Robotics, Motion and Machine Vision in the Automotive Industries, 1995, pp 487 – 492.

Mead C. A., and Mahowald M. A.

"A Silicon Model of Early Visual Processing", Neural Networks, Vol. 1, 1988, pp 91 – 97.

Menon, M. M., and Heinemann, K. G.

"Classification of Patterns Using a Self-Organising Neural Network", Neural Networks Vol. 1, 1988, pp 201 - 215.

Miclet, L.

"Grammatical Inference", Syntactic and Structural Pattern Recognition Theory and Application, 1990, ISBN 9971-50-566-5, pp 237 - 290.

Minsky, M. L., and Papert, S.

"Perceptrons", MIT Press, 1969.

Mohr, R.

"A General Purpose Line Drawing Analysis System", Syntactic and Structural Pattern Recognition Theory and Application, 1990, ISBN 9971-50-566-5, pp 479 - 497.

Newland, D. E.

"An introduction to Random Vibrations and Spectral Analysis", 1981, ISBN-0-582-46335-1.

Paige C. C., and Saunders M. A.

"Solution of Sparse Indefinite Systems of Linear Equations", SIAM J. Numer. Anal., Vol. 12, No. 4, September 1975.

Park, J., and Sandberg, I. W.

"Approximation and Radial-Basis-Function Networks", Neural Computation, Vol. 5, No. 2, March 1993.

Parker, D. B.

"Learning logic", 1982, Invention Report S81 - 64, File 1, Office of Technology Licensing, Stanford University, Stanford, CA.

Pearson, D. E., Hanna, E., and Martiniz, K.

"Computer-generated cartoons", Proceedings of the Rank Prize Funds Symposium on Images and Understanding. Royal Society, London. October 1986.

Perantonis, S. J., and Lisboa, J. G.

"Translation, Rotation, and Scale Invariant Pattern Recognition by High-Order Neural Networks and Moment Classifiers", IEEE Transactions on Neural Networks, Vol. 3, No. 2, March 1992.

Perantonis, S. J.

"Higher Order neural networks for invariant pattern recognition", Neural Networks current applications, P. J. G. Lisboa (Ed.), Pub. Chapman and Hall, ISBN 0 412 427907, 1992.

Poggio, T., and Christof, R.

"Synapses that Compute Motion", Scientific American, May 1987, Vol. 256, No. 5, pp 42 - 48.

Roberts, M. B. V.

"Biology a Functional Approach", Thomas Nelson and Sons Ltd, 1974, ISBN 0 17 143025 5.

Rosenblatt, F.

"The Perceptron: a probabilistic model of information storage and retrieval", Psychological Review, 1958, 65, pp 386 - 408.

Rosenblatt, F.

"Principles of Neurodynamics", Spartan Books, Washington DC, 1959.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J.

"Learning internal representations by error-propagation", Parallel distributed processing, 1986, Vol. 1, pp 318 - 362.

Sadovnik, L., Rashkovskiy, O., and Tebelev, I.

"Invariant wavelet transform-based automatic target recognition", 1995, SPIE, Vol. 2490, pp 179 – 185.

Sanfeliu, A.

"Matching Tree Structures", Syntactic and Structural Pattern Recognition Theory and Application, 1990, ISBN 9971-50-566-5, pp 145 - 178.

Shepherd, T. S., Uttal, W., Dayanand, S., and Lovell, R.

"A Method for Shift, Rotation, and Scale Invariant Pattern Recognition Using the Form and Arrangement of Pattern-Specific Features", Pattern Recognition, Vol. 25, No. 4, pp 343 - 356, 1992.

Spitzer, H., and Hochstein, S.

"Complex-cell receptive field models", Progress in Neurobiology, 1988, Vol. 31, pp 285 to 309.

Squire, D. M.

"Model-Based Neural Networks For Invariant Pattern Recognition", Ph.D. Thesis, Curtain University of Technology, 27 October 1996.

Stephenson, G.

"Mathematical Methods for Science Students", 1973, ISBN 0-582-44416-0, pp 376 - 377.

Suárez Arajo, C. P.

"Novel Neural Network Models for Computing Homothetic Invariances: An Image Algebra Notation", Journal of Mathematical Imaging and Vision 7, 1997, pp 69 – 83.

Tai, J. W., and Liu Y. J.

"Chinese Character Recognition", Syntactic and Structural Pattern Recognition Theory and Application, 1990, ISBN 9971-50-566-5, pp 415 - 451.

Tanaka, E.

"Parsing and Error-Correcting Parsing for String Grammars", Syntactic and Structural Pattern Recognition Theory and Application, 1990, ISBN 9971-50-566-5, pp 55 - 83.

Tanomaru, J., and Inubushi, A.

"A Compact Representation of Binary Patterns for Invariant Recognition", IEEE International Conference on Systems, Man and Cybernetics, 1995, pp 1550 – 1555.

Taylor, J. G.

"A Silicon Model of Vertebrate Retinal Processing", Neural Networks, Vol. 3, pp 171 – 178, 1990.

Thiaville L., Guillemaud R., and Niez J-J.

"Neural Networks for Edge Detection: A Performance Study", International Neural Network Conference, 1990.

Thomason, M. G.

"Introduction and Overview", Syntactic and Structural Pattern Recognition Theory and Application, 1990, ISBN 9971-50-566-5, pp 3 - 25.

Thompson, R. F.

"The Brain", W. H. Freeman and Company, 1985, ISBN 0-7167-1461-2, 0-7167-1462-0.

Tsang, P. W. M.

"A Genetic Algorithm for Affine Invariant Object Shape Recognition", Genetic Algorithms in Engineering Systems: Innovations and Applications, 13-14 September 1995, pp 293-298.

Uwechue, O. A., Pandya, A., and Szabo, P.

"High-Order Neural Networks for Image Recognition", 1995, SPIE Vol. 2568, pp 252 – 263.

Ventura, J. A, Chen, J.-M.

"A structural model for shape recognition using neural nets", Journal of Intelligent Manufacturing, 1996, Vol. 7, pp 1 –11.

Wang, D.

"An Extended Model of the Neocognitron for Pattern Partitioning and Pattern Composition", IEEE INNS International Joint Conference On Neural Networks, 1989 Vol. 2, pp II267 - II273.

Wasserman, P.

"Neural Computing: theory and practice", Van Nostrand Reinhold, 1989, ISBN 0-442-20743-3.

Wechsler, H.

"Invariance in Pattern Recognition", Advances in Electronics and Physics, Vol. 69, 1987, pp 261 - 322.

Weng, J., Ahuja, N., and Huang, T. S.

"Learning Recognition and Segmentation Using the Cresceptron", International Journal of Computer Vision 25(2), 1997, pp 109 – 143.

Werbos, P. J.

"Beyond regression: New tools for prediction and analysis in the behavioural sciences", 1974, Master's thesis, Harvard University.

Widrow, B.

"Generalization and information storage in networks of ADALINE neurons", in Yovitts G. T. (Ed.) 1962, Self-Organizing Systems, Spartan Books, Washington DC.

Widrow, B., and Hoff, M. E.

"Adaptive switching circuits", 1960 IRE WESCON Convention Record, New York: IRE, pp 96 - 104.

Widrow, B., and Winter, R.

"Neural Nets for Adaptive Filtering and Adaptive Pattern Recognition", Computer, March 1988, pp 25 - 39.

Wilson, S. W.

"Adaptive 'Cortical' Pattern Recognition", Proceedings of an International Conference on Genetic Algorithms and Their Applications, 1985, GrefenStette, J. J., ed., Hillsdale, NJ: Lawrence Erlbaum Assoc., pp 188 – 196.

Wong, E. K.

"Three-Dimensional Object Recognition by Attributed Graphs", Syntactic and Structural Pattern Recognition Theory and Application, 1990, ISBN 9971-50-566-5, pp 381 - 414.

Wood, J.

"Invariant Pattern Recognition: A Review", Pattern Recognition, Vol. 29, No. 1, 1996, pp 1 – 17.

Wood, J., and Shaw-Taylor, J.

"A unifying framework for invariant pattern recognition", Pattern Recognition Letters, 17 (1996), pp 1415 – 1422.

Yi, C.-H., Schlabbach, R., Kroth, H., and Klar, H.

"A New Bio-Inspired Algorithm for Early Vision Edge Detection and Spatial Segmentation", Biological and Artificial Computation: From Neuroscience to Technology IWANN 97, June 1997, pp 1107 – 1114.

Zhau, Q., and Bao, Z.

"Radar Target Recognition Using a Radial Basis Function Neural Network", Neural Networks, Vol. 9, No. 4 pp 709 - 720, 1996.

Zhou, Y.- T., and Chellappa, R.

Artificial Neural Networks for Computer Vision, 1992, Springer-Verlag, ISBN 0-387-97683-3.

Zhu, Z., Xi, H., and Xu, G.

"Combining Rotation-Invariance Images and Neural Networks for Road Scene Understanding", IEEE International Conference on Neural Networks ICNN 96, June 1996, pp 1732 – 1737.

Zwicke, P. E., and Kiss, I.

"A New Implementation of the Mellin Transform and its application to Radar Classification of Ships", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-5, No. 2, March 1983, pp 191 - 199.

APPENDIX 1 - DECISION BOUNDARY CALCULATION

This appendix explains the procedure used to calculate an approximate decision boundary for two neurons. A simplified model for the neuron is used to make it compatible with the synapse space representation. The simplifications from the standard neuron as described in Chapter 3 are:

- Maximum error correction is not used
- Conductance terms are not used
- An activation function is not used

Applying these simplifications to the neuron equations given by Equation 3.6 and Equation 3.47 gives Equation A1.1.

$$Out = \left(1 - \sqrt{\frac{\sum_{j=1}^{n}\left(w_j - x_j\right)^2}{n}}\right)(1-T) \qquad \text{A1.1}$$

For a competitive learning strategy, the boundary between two regions defining which learning will be selected (decision boundary) is defined as the locus of the points for which the output from the neurons is of equal magnitude. For two neurons, **a** and **b**, the boundary is given by the solution to Equation A1.2.

$$\left(1 - \sqrt{\frac{\sum_{j=1}^{n}\left(w_{aj} - x_j\right)^2}{n}}\right)(1-T_a) = \left(1 - \sqrt{\frac{\sum_{j=1}^{n}\left(w_{bj} - x_j\right)^2}{n}}\right)(1-T_b) \quad \text{A1.2}$$

Where the boundary is given by solving for x_j.

For the non-trivial case of $T_a \neq T_b$, the equation is difficult to solve analytically, so it is solved numerically. This can be solved by defining an error as the difference between the two sides of the equation and solving to give a zero error. Restricting the number of synapses to two, replacing x_1 by x and x_2 by y, splitting w_j into x and y components and rearranging can give the following error as function of x.

$$f(x) = \left(\sqrt{n} - \sqrt{(x-x_a)^2 + (y-y_a)^2}\right)(1-T_a) - \left(\sqrt{n} - \sqrt{(x-x_b)^2 + (y-y_b)^2}\right)(1-T_b) \quad \text{A1.3}$$

Another way of rewriting the equation is given by Equation A1.4.

$$g(x) = \left(\sqrt{n} - \sqrt{(x-x_a)^2 + (y-y_a)^2}\right) - \left(\sqrt{n} - \sqrt{(x-x_b)^2 + (y-y_b)^2}\right)\frac{(1-T_b)}{(1-T_a)} \quad \text{A1.4}$$

Equation A1.4 has the useful property combining the tiredness terms into one quantity called the relative alertness, with values ranging from 0 (completely tired) to infinity (other neuron is completely tired). Because of the possibility of a division by zero, it was not used to solve the decision boundary equation.

Equation A1.3 can be solved numerically by Newton's method [Stephenson 1973], this can be stated.

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} \quad \text{A1.5}$$

For an estimate of the solution for $f(x) = 0$ at iteration k, a better estimate can be obtained for the value at iteration k + 1. If the estimate for the solution is too far away from the solution, it is possible for this method to fail because it will not converge. For each iteration the initial value for x was given as the uncompensated (does not use tiredness) decision boundary x co-ordinate. This was found to produce stable iterations.

Differentiating A1.3 with respect to x gives

$$f'(x) = -\frac{(x-x_a)(1-T_a)}{\sqrt{(x-x_a)^2+(y-y_a)^2}} + \frac{(x-x_b)(1-T_b)}{\sqrt{(x-x_b)^2+(y-y_b)^2}} \qquad \text{A1.6}$$

Care must be taken when differentiating Equation A1.3 because of sign changes caused by the square and square root functions. Rewriting Equation A1.3 with $y = y_a$, and $y = y_b$ gives

$$f(x)\big|_{y=y_a,y=y_b} = \left(\sqrt{n} - \sqrt{(x-x_a)^2}\right)(1-T_a) - \left(\sqrt{n} - \sqrt{(x-x_b)^2}\right)(1-T_b) \qquad \text{A1.7}$$

It is tempting to cancel out the square root and square functions in Equation A1.7, but as these functions represent distance, the square root function must always return a positive quantity. However, the square root and square functions can be replaced by the modulus function. Considering the part of equation due to neuron **a** gives a maximum value at position $x = x_a$, therefore the gradient is positive for $x < x_a$ and negative for $x > x_a$. Similarly the part of equation due to neuron **b** gives a minimum value at position $x = x_b$, therefore the gradient is negative for $x < x_b$ and positive for $x > x_b$. These gradient conditions are obeyed by Equation A1.6.

The decision boundary equation can be solved using Equations A1.3, A1.5, and A1.6 with the termination criterion being the absolute error that must be less than a defined tolerance. An Excel spreadsheet running on an IBM-compatible computer was used to solve the equation because the calculated decision boundary could easily be displayed graphically.

A simple Excel Basic program was used to iterate a solution, with the cells providing the calculation parameters and acting as the repository for the data. Selecting the output rows and columns of data defining the boundary and then selecting the charting button produces a graphical display.

Two lines are drawn on the chart, a dashed line indicating the uncompensated decision boundary and the full line indicating the tired decision boundary.

The following figures show the effects of different tiredness values of the decision boundary between two neurons with the following characteristics:

Number of synapses	2
Range of synapse values	0 to 1.0
Neuron **a** co-ordinates	$x = 0, y = 0.5$
Neuron **b** co-ordinates	$x = 0.2, y = 0.5$

The values were calculated using an increment in the values for y of 0.1 giving 11 co-ordinates for the decision boundary.

Figure A1.1 is for $T_a = 0$ and $T_b = 0$, giving a relative alertness of 1.0, which represents the initial starting conditions of the learning process and is shown by a vertical line at $x = 0.1$. This is the same locus as the uncompensated decision boundary locus.

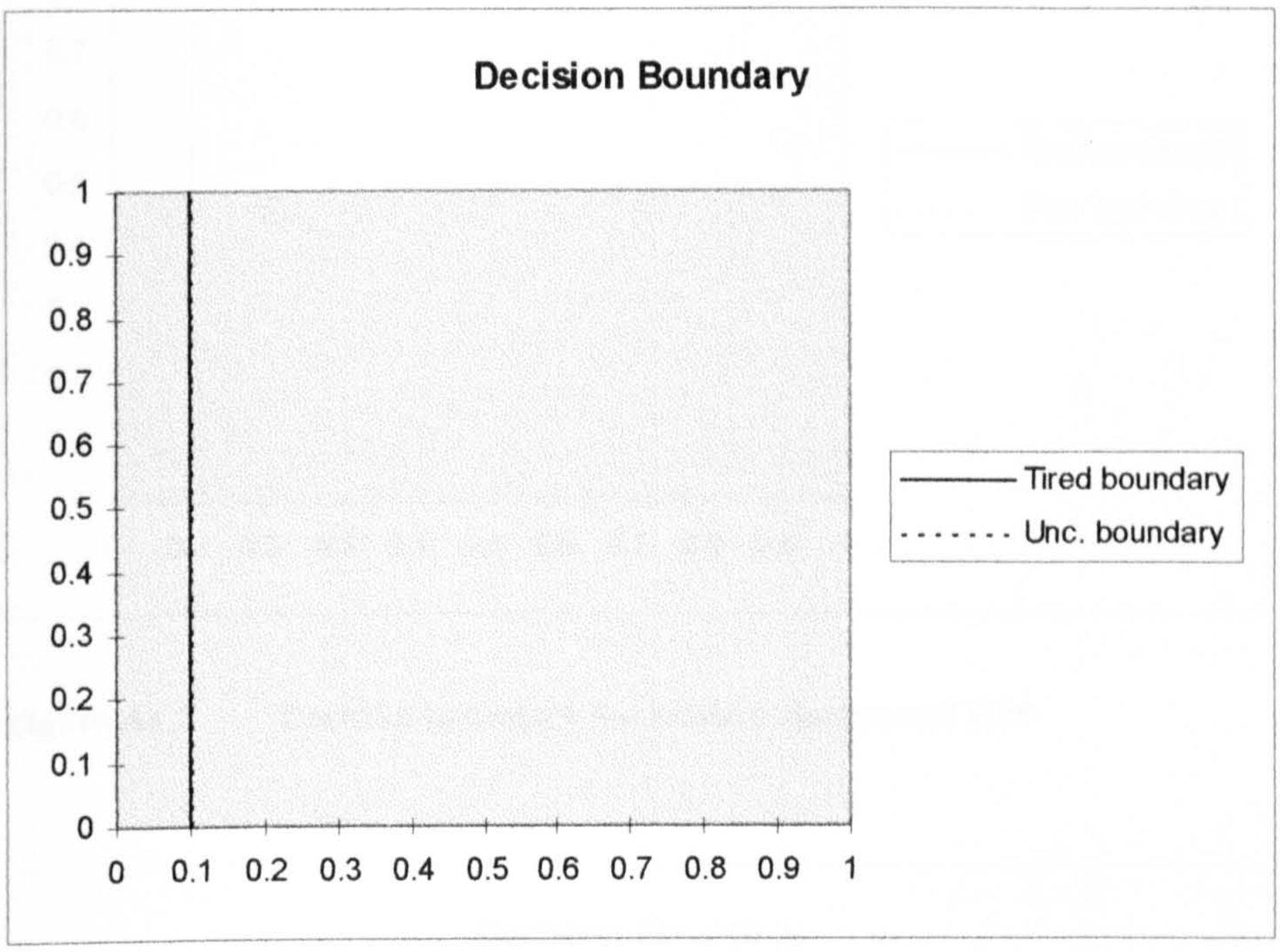


Figure A1.1 Decision boundary for relative alertness of 1.0

Figure A1.2 shows the decision boundary for $T_a = 0$ and $T_b = 0.01$, giving a relative alertness of 0.99. This represents the situation where neuron **b** has been taught and the tiredness has slightly increased. There has been a slight change in the decision boundary causing it to move towards neuron **b**.

Slightly increasing the tiredness of neuron **b** to 0.05 will give a relative alertness of 0.95, and this is shown in Figure A1.3. The effect of the tiredness is much greater than before with the decision boundary for $y = 0$ and $y = 1.0$ further to the right than the position of neuron **b**.

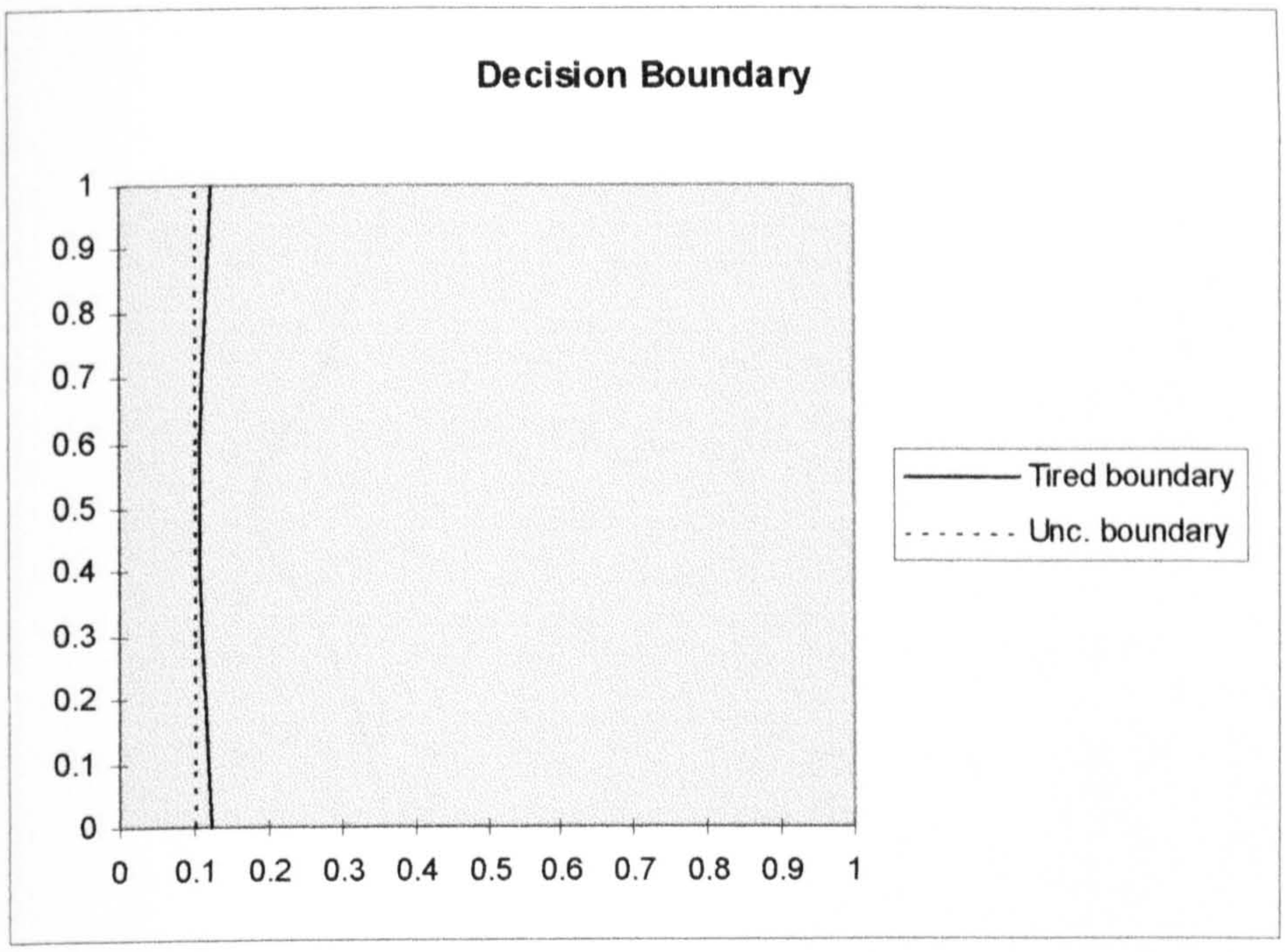


Figure A1.2 Decision boundary for relative alertness of 0.99

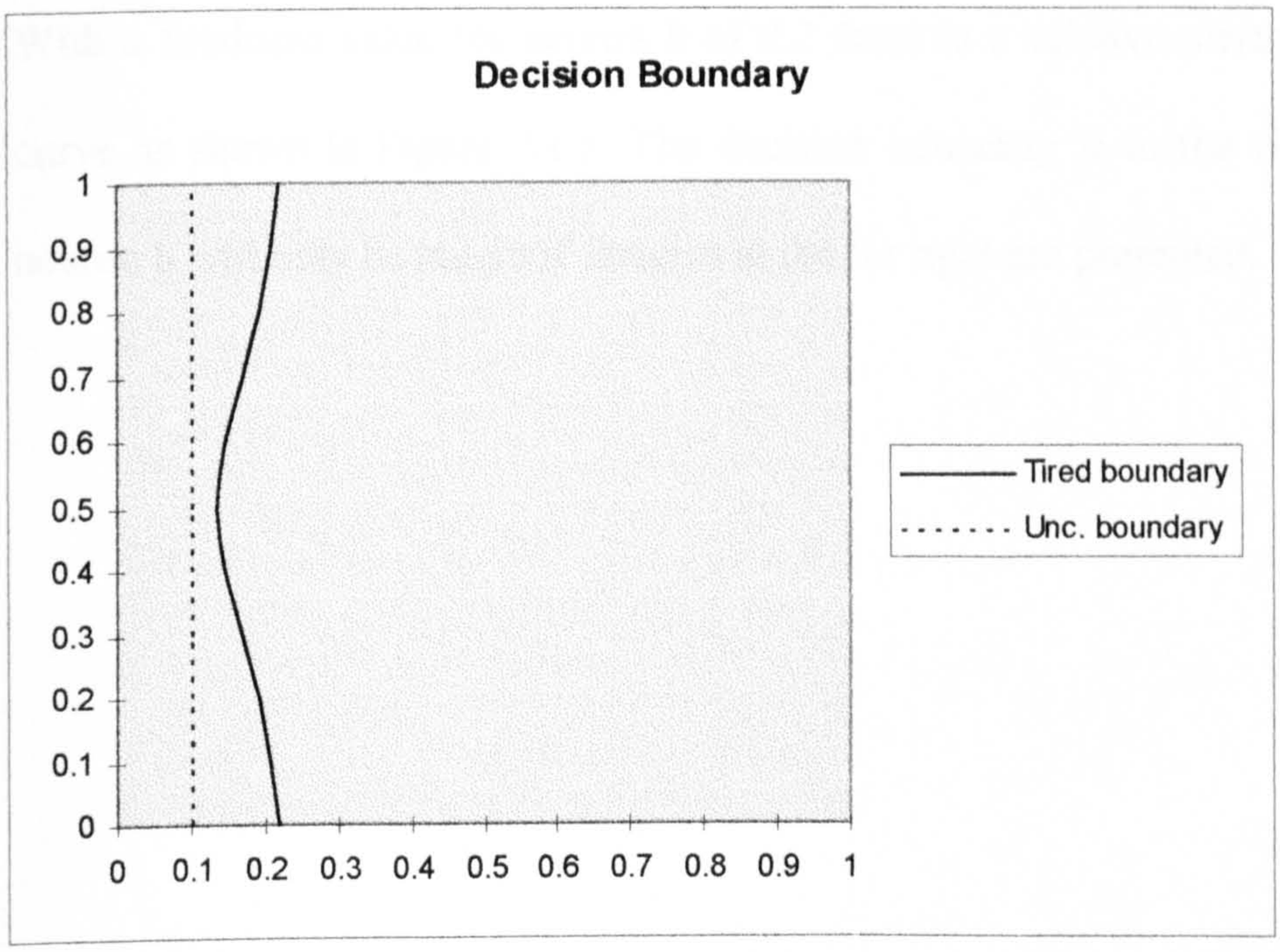


Figure A1.3 Decision boundary for relative alertness of 0.95

Further increasing the tiredness of neuron **b** to 0.1 gives a relative alertness of 0.9 with a decision boundary shown in Figure A1.4. The decision boundary just encloses neuron **b**.

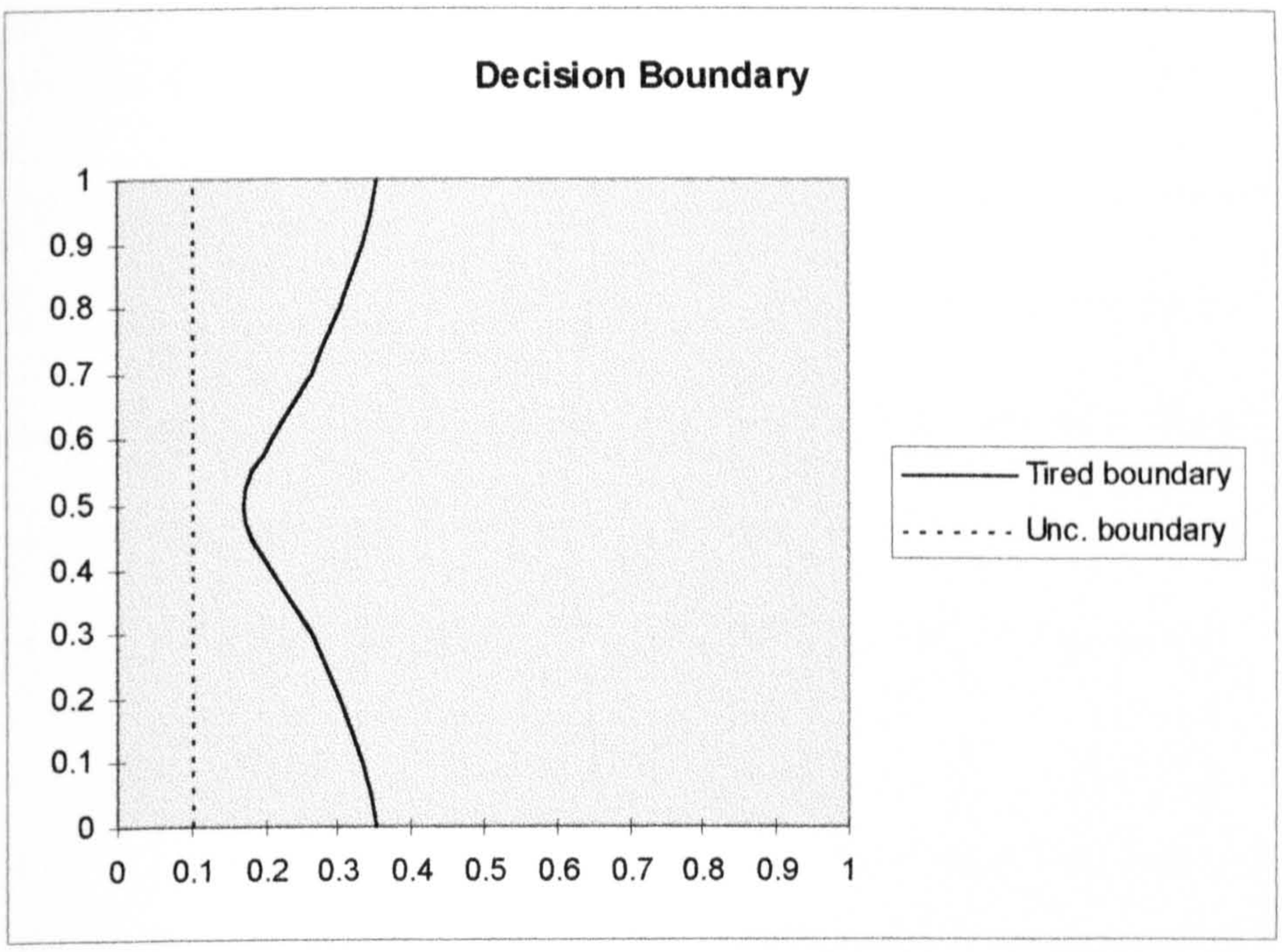


Figure A1.4 Decision boundary for relative alertness of 0.9

With a tiredness value for neuron **b** of 0.2 there is a relative alertness of 0.8 giving the curve as shown in Figure A1.5. The decision boundary is to the right of neuron **b**, and neuron **b** will only be taught if features at the far right are presented.

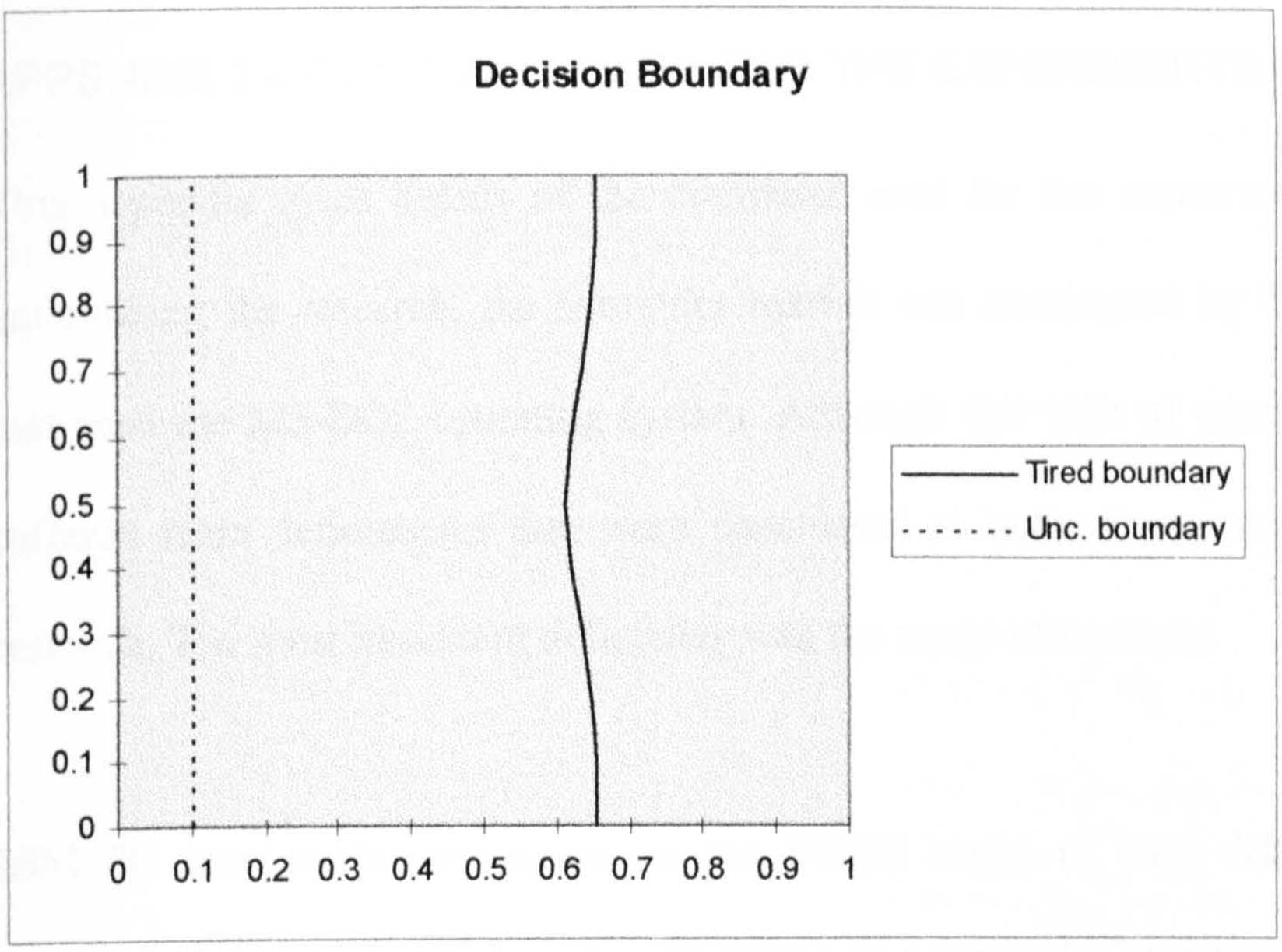


Figure A1.5 Decision boundary for relative alertness of 0.8

Further increases in the tiredness of neuron **b** will shift the decision boundary to the right.

APPENDIX 2 – COMPUTER USED FOR THE EXPERIMENTS

This appendix gives details of the computer used for the experiments. At the time of undertaking the research, the computer market was dominated by IBM PC compatibles that used the MS-DOS operating system. Although this type of computer was popular, it suffered from deficiencies that were considered to make it unsuitable for this type of research. The most important deficiency was the usage of memory.

IBM PC compatibles are based on the 80X86 range of Intel Microprocessors. Early versions of the processor (less than 80386) had a method of addressing large amounts of memory that was rather primitive when compared with other contemporary microprocessors, like the Motorola 68000. The Intel microprocessors addressed the memory in 64 KB segments, if more memory was required the microprocessor addressed memory in a different segment using segment and offset registers. Microprocessors that do not have this type of architecture and have an address bus that can fully address the memory are deemed to have a linear memory space. The addressing range of the Intel microprocessors was also deemed to be poor, as illustrated in Table A2.1.

8086/80186	80286	80386 DX	80386 SX	68000	68020
1 MB Segmented	16 MB Segmented	4 GB Linear	16 MB Linear	16 MB Linear	4 GB Linear

Table A2.1 Comparison of memory capabilities of popular microprocessors

The marriage of Intel processors and the MS-DOS operating system became an industry standard. This was a double-edged sword. There were many different computer suppliers producing a vast range of machines and software packages. But as the primary market was commerce, commercial interests became paramount and, in order to justify the investment

in IT, newer PC compatible computers still had to be compatible with the older software packages. This lead to an innate conservatism in the market, in which newer microprocessors had to emulate earlier microprocessors, that stifled the development of 32-bit operating systems.

The original architecture of the IBM PC compatible had 640KB of RAM and the rest of the 1MB memory space being used for ROM and other facilities. Later microprocessors allowed various methods of adding extra memory to the computers. The 640KB of memory became known as base or conventional memory, and the extra memory became known as extended or expanded memory.

How does this affect neural network research? One feature of neural network programs is that they use large arrays to represent the neurons in a network. When developing programs to run on MS-DOS based computers, different parts of a program are stored in different memory segments. After spending many fraught hours at work trying to reduce the size of programs so that they would work under MS-DOS, it was considered that the choice of computer needed to be carefully made. One method of writing large programs on MS-DOS based computers is to use a DOS extender, which allows programs written in 80286 or 80386 machine code to run under MS-DOS. Normally these tools are bought in addition to the basic compiler and it might be necessary to buy a specialist debugger.

As well as software development, analysis of the results is also an important issue. Editors would be required to browse large log files, some of which could be hundreds of KB long, also the ability to view multiple files would be a very useful facility.

The file browsing facilities tended to favour a computer with a graphical user interface (GUI). At the time of choosing, the computer GEM was quite widely used on PCs and Windows version 2 was also available. Other computers that were available at that time having a linear address space and a GUI were;

- Apple Macintosh
- Atari ST
- Commodore Amiga
- Acorn Archimedes

The first three used Motorola based 68000 series computers, whilst the Acorn Archimedes used an in-house design 32-bit RISC (Reduced Instruction Set Computer). The Apple Macintosh was a business rival to the PC; it was expensive, had a very good reputation and had dominance of the DTP market. Work-related contacts with Apple Centres about software development seemed to bear little fruit. The sales force knew about DTP and nothing else. The Atari ST also used the GEM operating system and because of the built-in MIDI port, it was extensively used in the music industry. However, it had a reputation for being a toy computer. Also having a reputation for being a toy computer was the Commodore Amiga, but because of its ability to display up to 4096 colours, it was professionally used for computer graphics. Both the Atari ST and the Commodore Amiga were difficult to expand because of their keyboard style cases.

The Acorn Archimedes fell someway between the Apple Macintosh and the other computers. It was extensively used in education because of its compatibility with the old Acorn BBC. Probably because of the its British origins, there was a home grown industry

supporting the computer with both hardware and software products available at reasonable prices.

It was decided to buy the Acorn Archimedes. It came with a modest specification but was expanded during the lifetime of the research. Table A2.2 lists the computer specifications.

Status	Processor	RAM	Hard disk
Original	ARM2 8MHz	1MB	-
Upgraded	ARM3 25MHz	4MB	80MB

Table A2.2 Acorn Archimedes specifications

After choosing the computer, it was necessary to choose the programming language. After using various programming languages professionally, the Pascal/Modula-2 approach was preferred because of the strong type checking. The company that was planning to release a version of Modula-2 turned out to only produce vapourware. A Pascal compiler was bought but it turned out to be not suited for serious development, the string and file handling was poor and it would not support modular compilation although other source files could be included into the main program. After suffering these disappointments, it was decided to buy and use a C-compiler.

APPENDIX 3 – SHIFT INVARIANT NETWORK RESULTS

This appendix gives the results for tests carried out on the shift invariant network. Two different network topologies were tested, using a single layer network and a triple layer network.

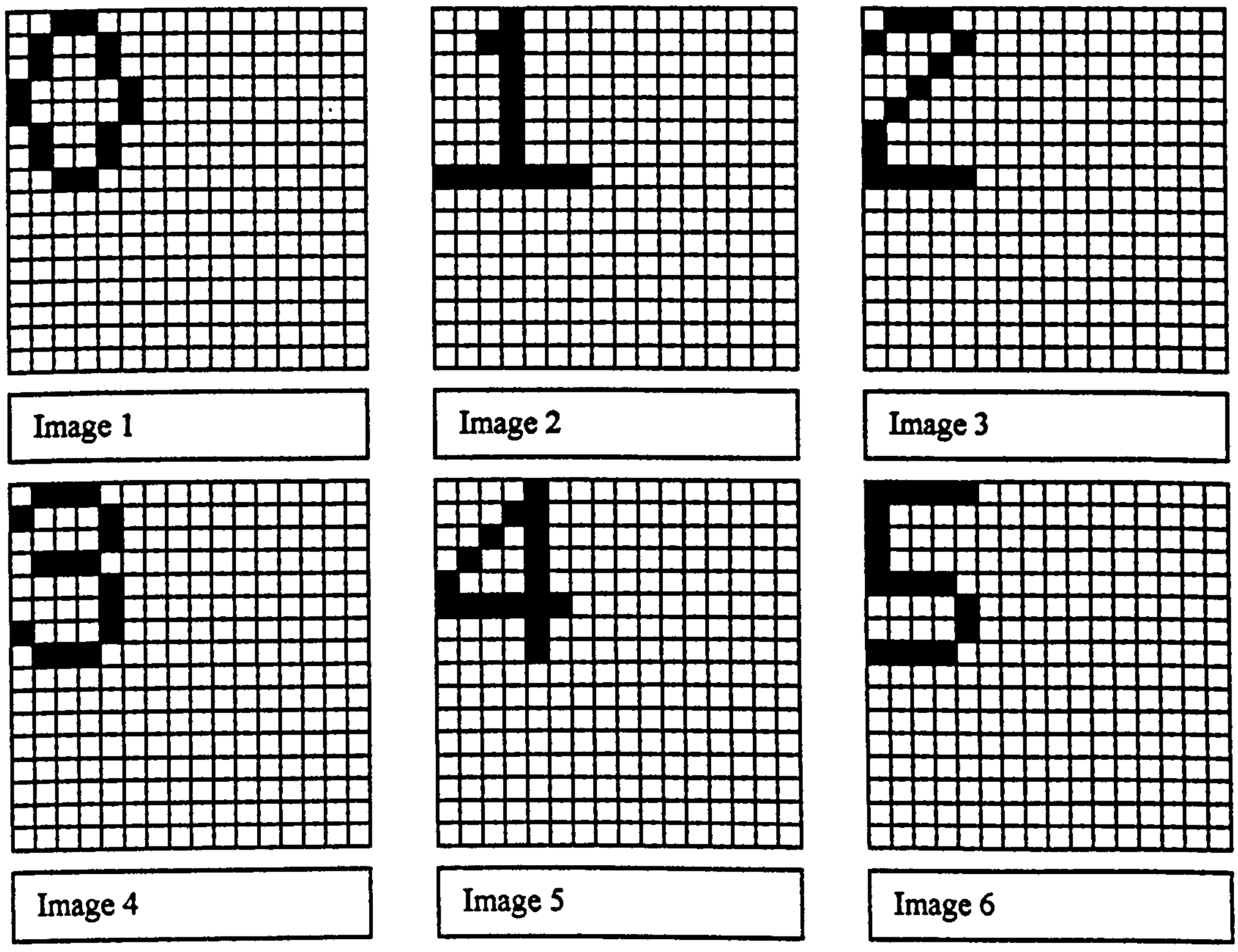
SINGLE LAYER TESTING

When testing the single layer the network had the following characteristics:

Layer Number	Number of Features	Receptive Field Rows	Receptive Field Columns
1	11	8	8

Table A3.1 Single layer network configuration

The test images had a size of 16 rows by 16 columns, and are shown in Figure A3.1.



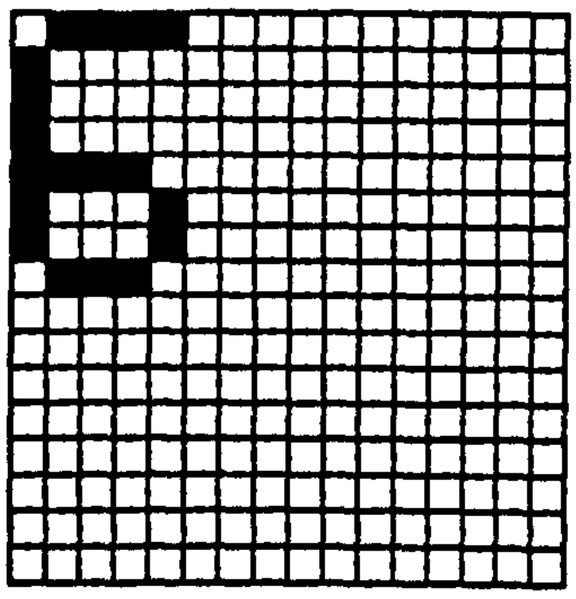


Image 7

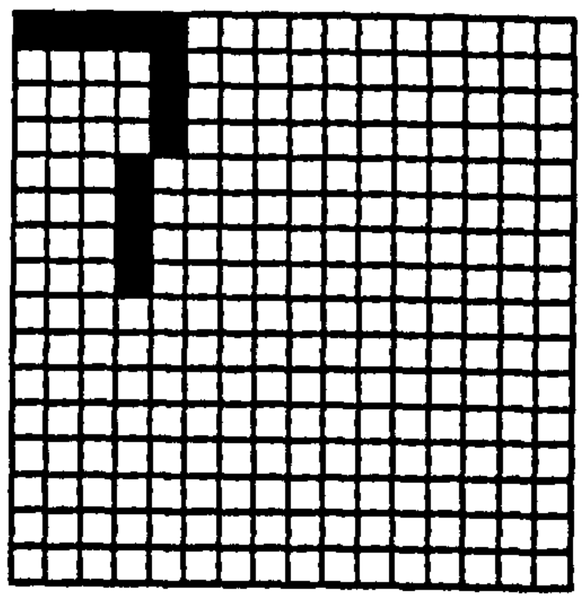


Image 8

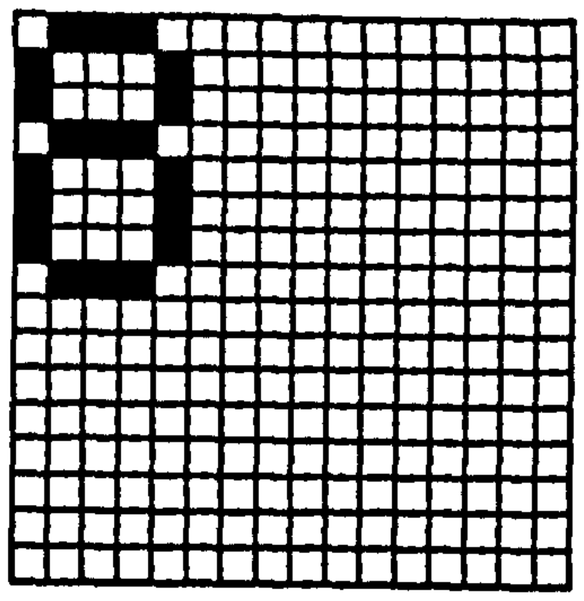


Image 9

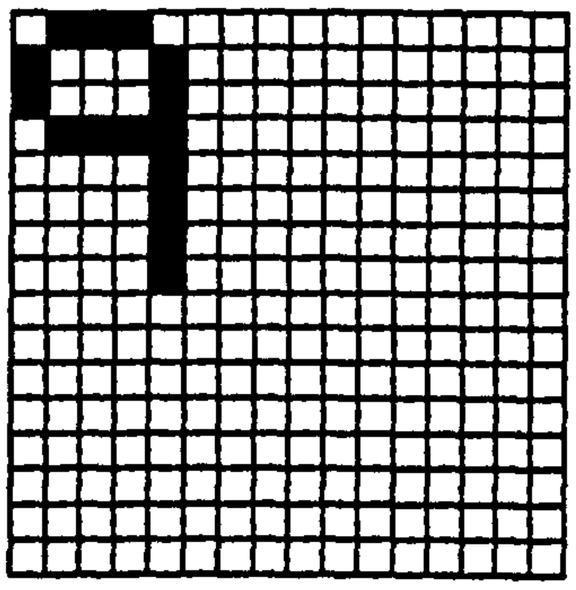


Image 10

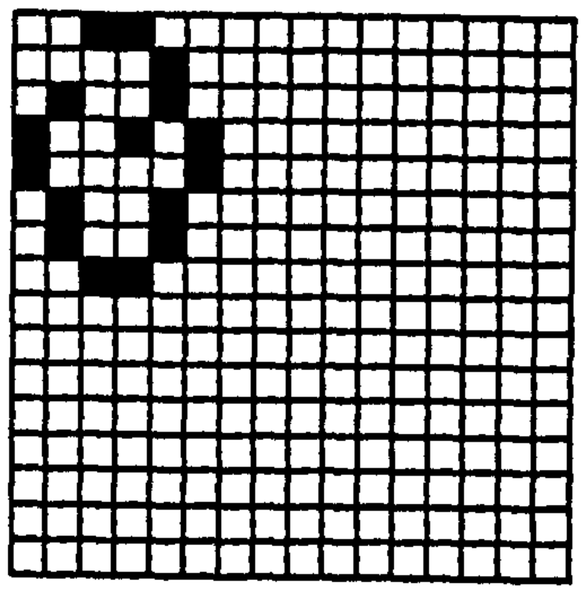


Image 11

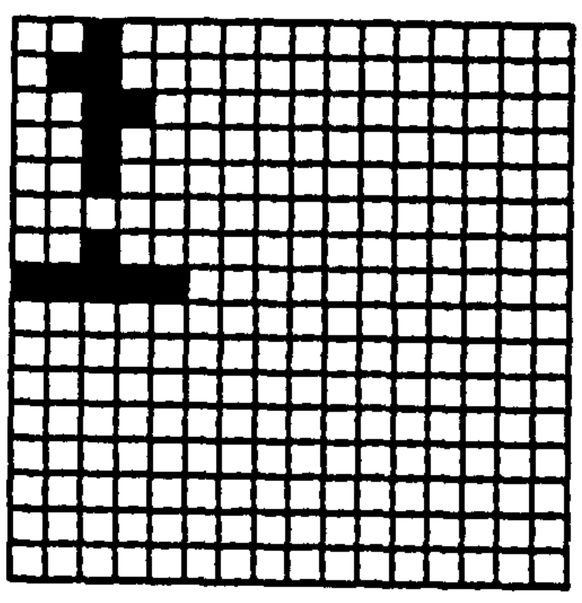


Image 12

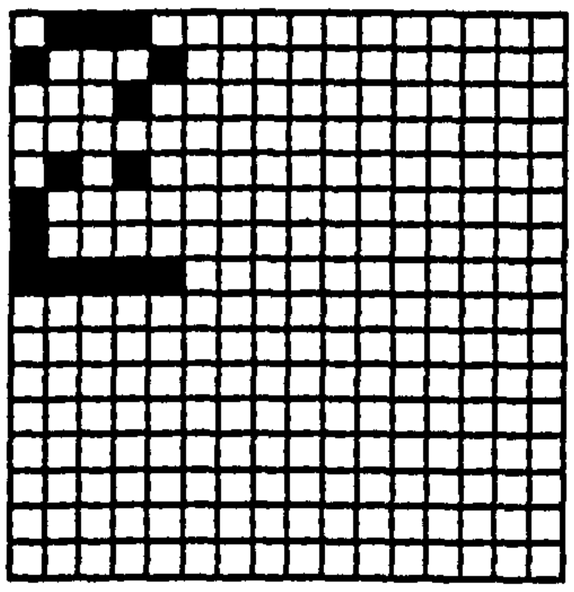


Image 13

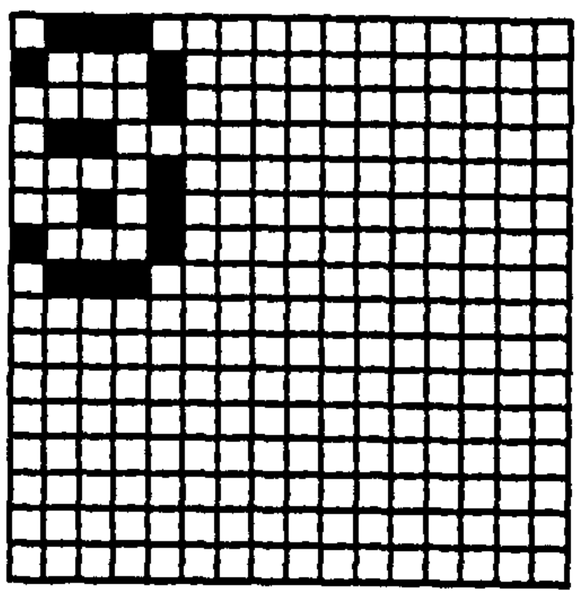


Image 14

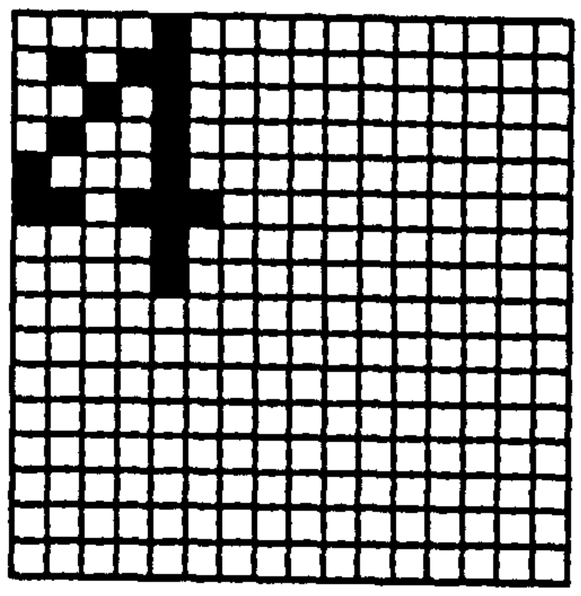


Image 15

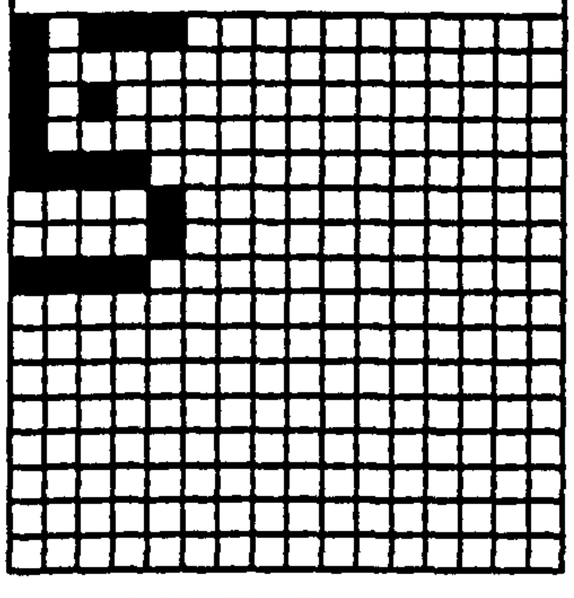


Image 16

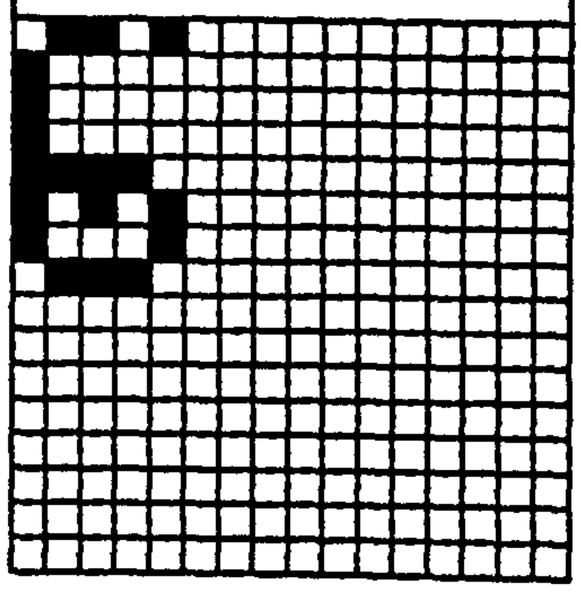


Image 17

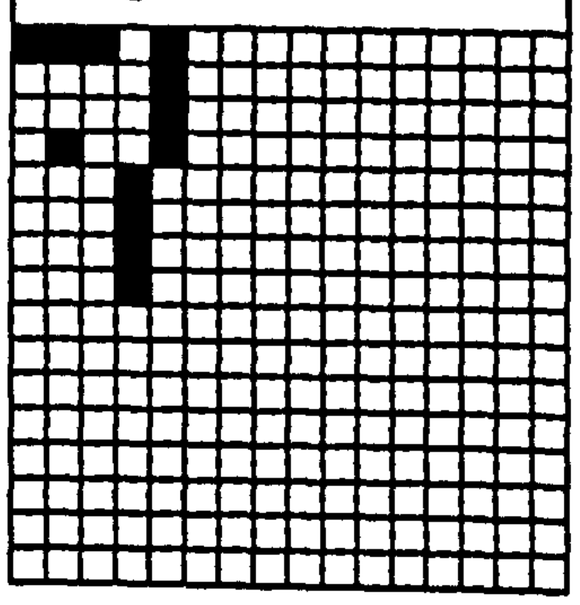


Image 18

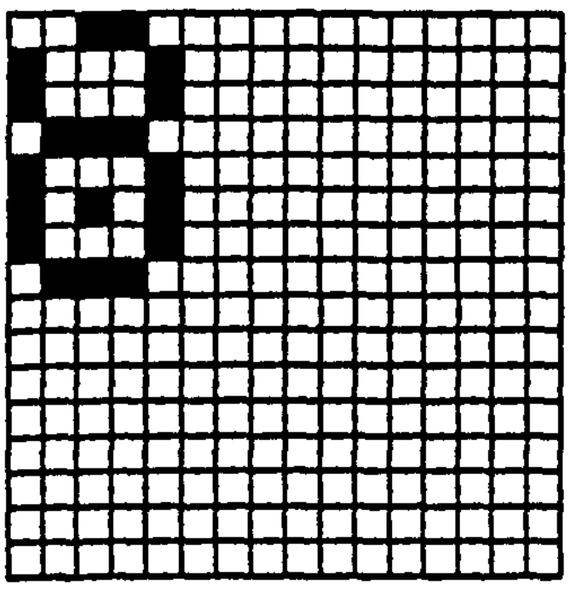


Image 19

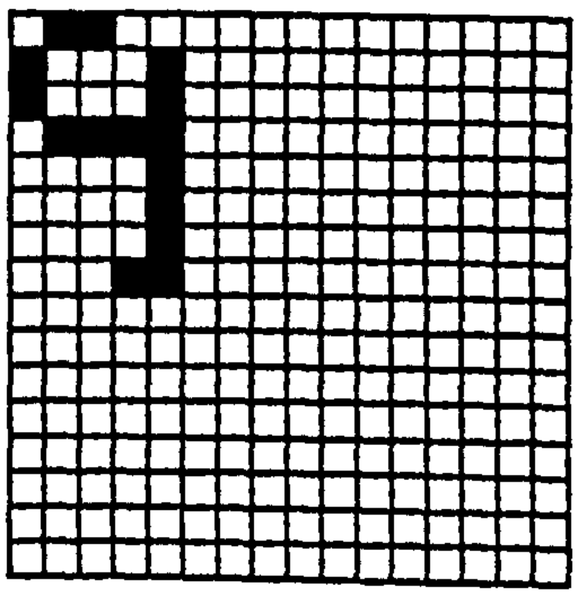


Image 20

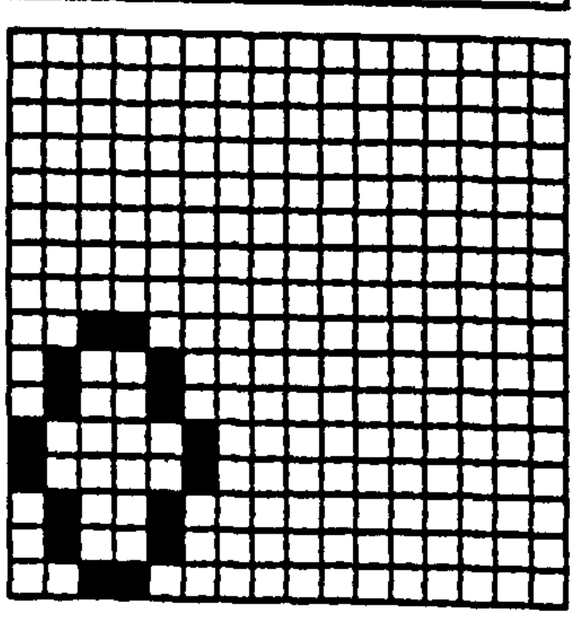


Image 21

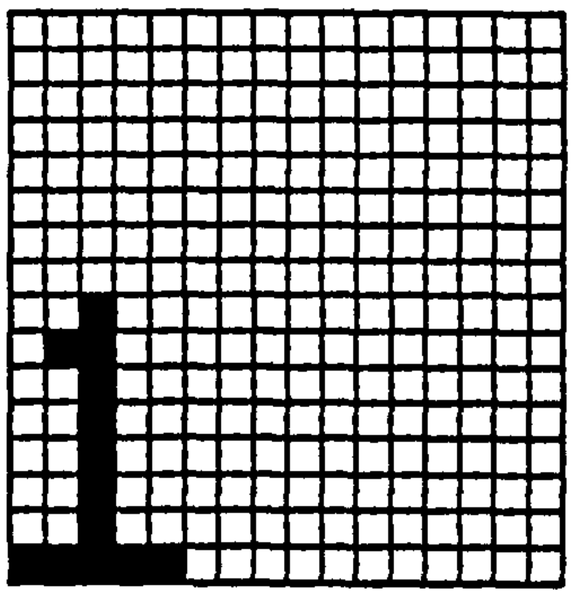


Image 22

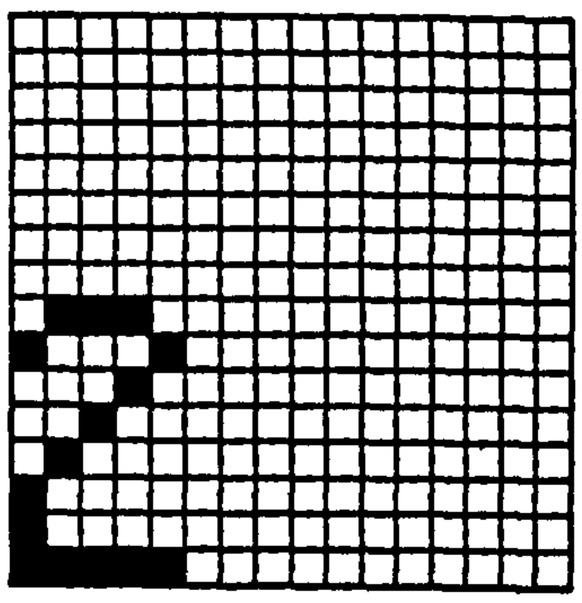


Image 23

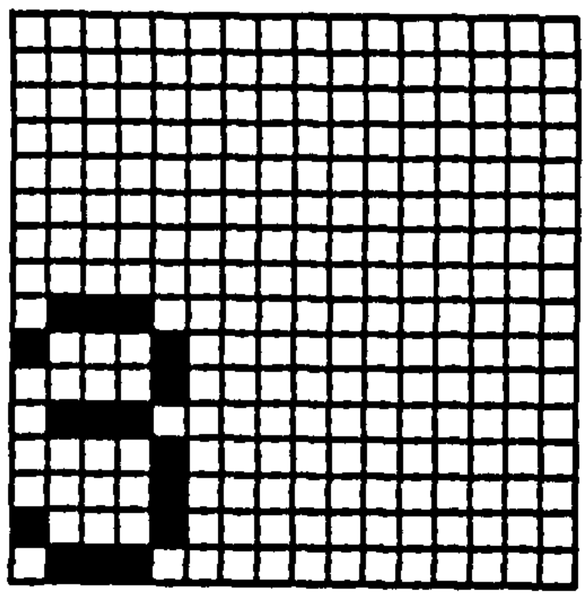


Image 24

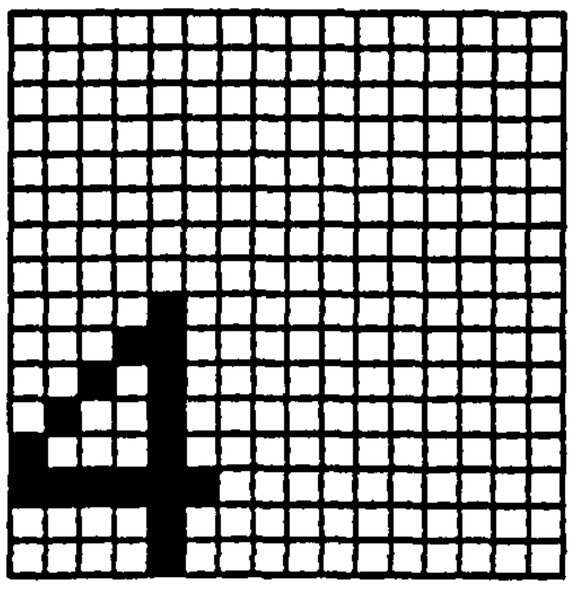


Image 25

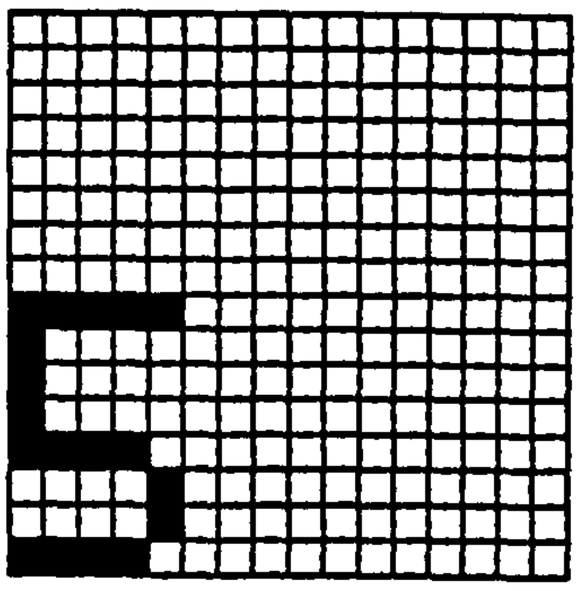


Image 26

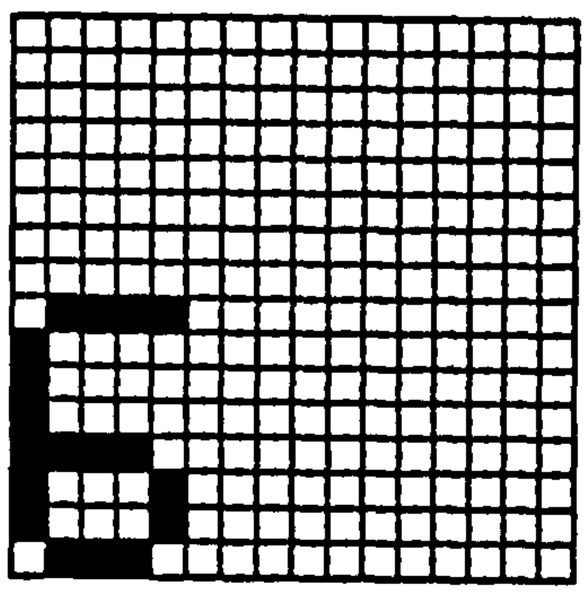


Image 27

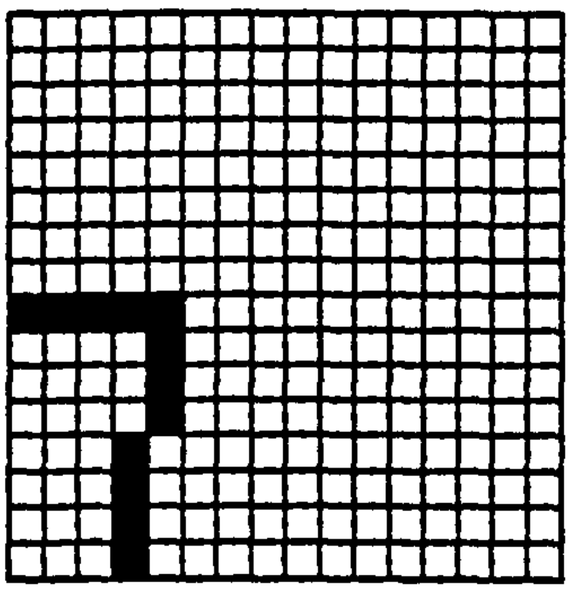


Image 28

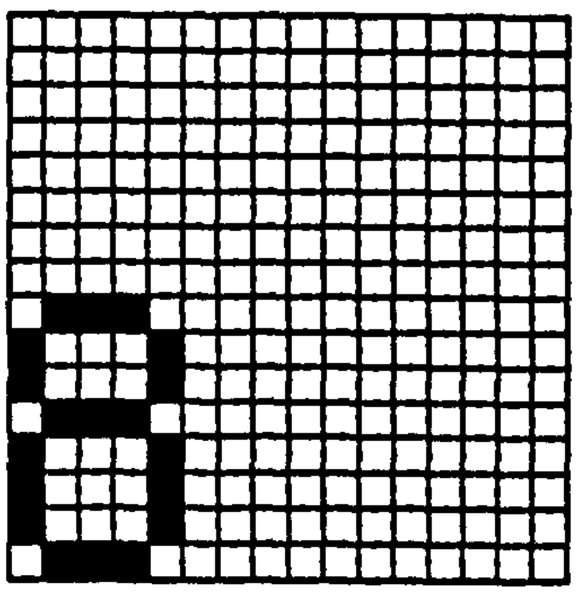


Image 29

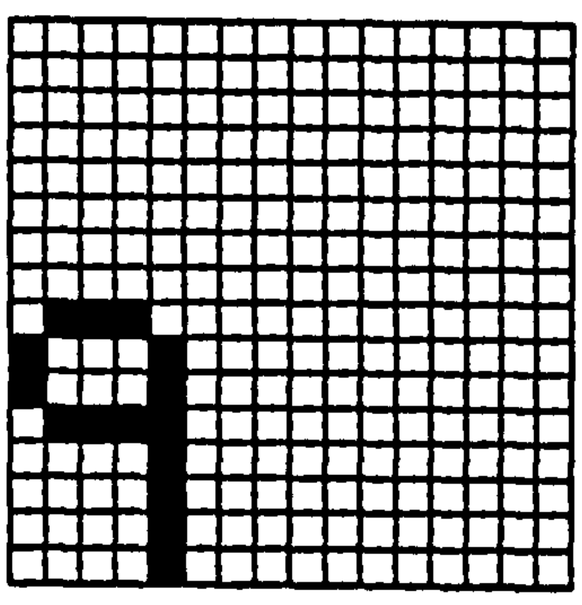


Image 30

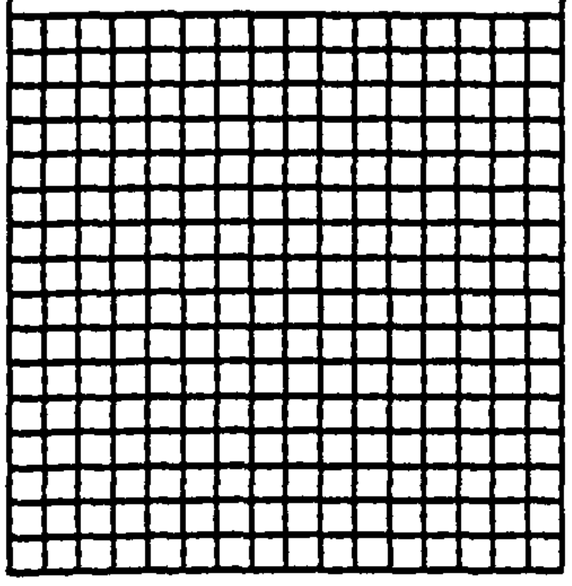


Image 31

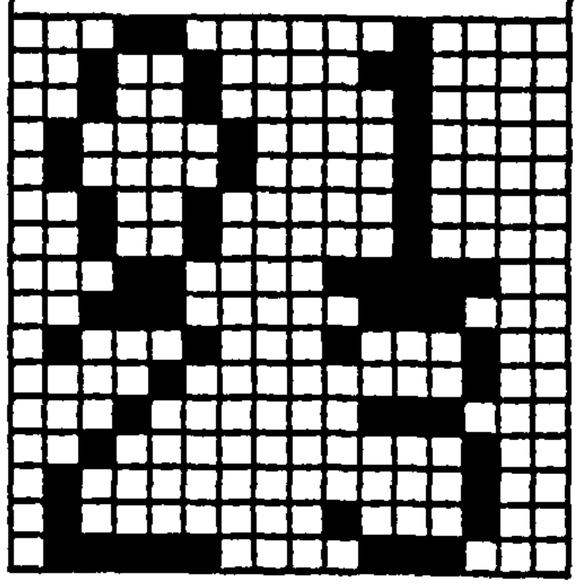


Image 32

Figure A3.1 Image set used

A dark coloured pixel has a value of 1.0 and a light coloured pixel has a value of 0.0. Images 1 to 10 show the numbers 0 to 9 located in the top left quadrant of the retina. Noisy

versions of these images are shown in Images 11 to 20. The numbers in Images 1 to 10 are translated to the bottom left quadrant in Images 21 to 30. Image 31 only shows the background, and Image 32 shows the numbers 0, 1, 2 and 3. The symmetrical nature of neuron means that the background can be classified as a valid feature.

The teach file presented the network with the images in the sequence:

1. Image 1
2. Image 2
3. Image 3
4. Image 4
5. Image 5
6. Image 6
7. Image 7
8. Image 8
9. Image 9
10. Image 10
11. Image 31

The location of the hot spot was given as 12.5 rows and 4.5 columns for Images 1 to 10 and at location 4.5 rows and 4.5 columns for Image 31.

The following teaching and neuron run parameters were used:

Layer	Tiredness reduction	Tiredness enhancement	Atrophy	Tolerance	Learning rate	Teaching presentations
1	0.02	0.75	0.25	0.25	0.5	25

Table A3.2 Single layer network teaching and neuron run parameters

The number of teaching presentations is low compared with other types of neuron and network configuration. This was because the high learning rate would rapidly teach a neuron.

The total number of neurons in the network can be determined using Equations 3.18, 3.19 and 3.20.

$$\text{Number of neurons} = (16 - 8 + 1) \times (16 - 8 + 1) \times 11 = 891$$

There are 64 synapses for each neuron; this value is given by the size of the receptive field (8×8). The total number of synapses is 57024, given by the product of the two previous values.

For a non-shift invariant network, the receptive field would expand to fill the retina giving 11 neurons with each having 256 synapses. Hence there will be 2816 synapses in total.

On comparing the two networks, it is evident that on the forward classification route, the shift invariant will have about 20 times as many synapse calculations to perform, and there will be a corresponding time penalty.

This time penalty was reduced by defining a hot spot for the image and reduced the size of the image that needed to be examined during the training. This effectively reduced the problem to examining 11 neurons with each having 64 synapses, hence 704 synapses in total. To summarise, there were 11 neuron outputs to calculate on a training run and 891 neurons on a classification run.

On examination of the teach file and teaching run parameters, there are 25 presentations of 11 images giving a total of 275 presented images.

The system produced two types of files, a synapse file and an output file containing the following data:

- Synapse file

  This file contains the synapse values for each neuron and layer, and the conductance values for layer.

- Output file

  The output file contains the output values when an image is presented in three ways: the feature neuron number with the largest value for every neuron position in a layer, the maximum neuron output value for every neuron position and the neuron output value for a specified neuron at every neuron position. All output formats produce a rectangle of values.

As the full results for all 32 images would take up many pages, only the full results for images 1 and 32 are respectively given in Results A3.1 and Results A3.2.

```
Layer 1 image number 1
  3  1  8  8  8  8  8  8  8
  8  7  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
```

```
Layer 1 image number 1
1.000 0.368 0.444 0.508 0.594 1.000 1.000 1.000 1.000
0.368 0.391 0.508 0.548 0.594 1.000 1.000 1.000 1.000
0.444 0.444 0.548 0.594 0.652 1.000 1.000 1.000 1.000
0.508 0.508 0.594 0.652 0.729 1.000 1.000 1.000 1.000
0.548 0.548 0.652 0.729 0.863 1.000 1.000 1.000 1.000
0.594 0.594 0.729 0.863 1.000 1.000 1.000 1.000 1.000
0.729 0.729 0.863 1.000 1.000 1.000 1.000 1.000 1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
```

```
Layer 1 image number 1 feature number 1
 0.23  0.37  0.26  0.26  0.37  0.29  0.33  0.33  0.33
 0.21  0.23  0.29  0.35  0.29  0.29  0.33  0.33  0.33
 0.26  0.23  0.35  0.33  0.24  0.33  0.33  0.33  0.33
 0.23  0.29  0.33  0.27  0.29  0.33  0.33  0.33  0.33
 0.27  0.31  0.27  0.29  0.35  0.31  0.33  0.33  0.33
 0.33  0.29  0.29  0.31  0.33  0.33  0.33  0.33  0.33
 0.29  0.29  0.35  0.37  0.31  0.33  0.33  0.33  0.33
 0.37  0.37  0.33  0.31  0.33  0.33  0.33  0.33  0.33
 0.33  0.33  0.33  0.33  0.33  0.33  0.33  0.33  0.33
```

```
Layer 1 image number 1 feature number 2
 0.27  0.31  0.21  0.19  0.35  0.31  0.27  0.27  0.27
 0.26  0.24  0.24  0.26  0.27  0.31  0.27  0.27  0.27
 0.27  0.24  0.26  0.27  0.26  0.31  0.27  0.27  0.27
 0.24  0.31  0.27  0.26  0.31  0.27  0.27  0.27  0.27
 0.17  0.26  0.26  0.27  0.33  0.26  0.27  0.27  0.27
 0.24  0.24  0.27  0.29  0.27  0.27  0.27  0.27  0.27
 0.27  0.24  0.29  0.31  0.26  0.27  0.27  0.27  0.27
 0.31  0.31  0.27  0.26  0.27  0.27  0.27  0.27  0.27
 0.27  0.27  0.27  0.27  0.27  0.27  0.27  0.27  0.27
```

```
Layer 1 image number 1 feature number 3
 1.00  0.16  0.16  0.31  0.21  0.35  0.31  0.31  0.31
 0.29  0.24  0.24  0.26  0.27  0.31  0.31  0.31  0.31
 0.16  0.35  0.26  0.21  0.37  0.27  0.31  0.31  0.31
 0.24  0.24  0.27  0.33  0.31  0.27  0.31  0.31  0.31
 0.33  0.20  0.29  0.39  0.26  0.29  0.31  0.31  0.31
 0.27  0.27  0.31  0.29  0.27  0.31  0.31  0.31  0.31
 0.24  0.31  0.33  0.27  0.29  0.31  0.31  0.31  0.31
 0.35  0.31  0.27  0.29  0.31  0.31  0.31  0.31  0.31
 0.31  0.31  0.31  0.31  0.31  0.31  0.31  0.31  0.31
```

```
Layer 1 image number 1 feature number 4
 0.31  0.31  0.24  0.21  0.31  0.27  0.31  0.31  0.31
 0.23  0.21  0.27  0.29  0.27  0.27  0.31  0.31  0.31
 0.27  0.27  0.26  0.27  0.23  0.31  0.31  0.31  0.31
 0.27  0.24  0.27  0.29  0.27  0.31  0.31  0.31  0.31
 0.33  0.29  0.29  0.27  0.33  0.29  0.31  0.31  0.31
 0.27  0.27  0.27  0.29  0.31  0.31  0.31  0.31  0.31
 0.27  0.27  0.33  0.35  0.29  0.31  0.31  0.31  0.31
 0.35  0.35  0.31  0.29  0.31  0.31  0.31  0.31  0.31
 0.31  0.31  0.31  0.31  0.31  0.31  0.31  0.31  0.31
```

Layer 1 image number 1 feature number 5
0.26 0.29 0.23 0.23 0.33 0.29 0.33 0.33 0.33
0.27 0.26 0.26 0.27 0.29 0.33 0.33 0.33 0.33
0.37 0.29 0.27 0.29 0.24 0.37 0.33 0.33 0.33
0.33 0.29 0.29 0.31 0.33 0.33 0.33 0.33 0.33
0.35 0.35 0.31 0.29 0.39 0.31 0.33 0.33 0.33
0.29 0.29 0.33 0.35 0.33 0.33 0.33 0.33 0.33
0.29 0.29 0.35 0.37 0.31 0.33 0.33 0.33 0.33
0.37 0.37 0.33 0.31 0.33 0.33 0.33 0.33 0.33
0.33 0.33 0.33 0.33 0.33 0.33 0.33 0.33 0.33

Layer 1 image number 1 feature number 6
0.23 0.23 0.42 0.37 0.26 0.33 0.37 0.37 0.37
0.21 0.20 0.37 0.35 0.26 0.33 0.37 0.37 0.37
0.23 0.23 0.35 0.33 0.31 0.33 0.37 0.37 0.37
0.26 0.26 0.33 0.35 0.33 0.33 0.37 0.37 0.37
0.31 0.27 0.35 0.37 0.31 0.35 0.37 0.37 0.37
0.33 0.29 0.37 0.35 0.33 0.37 0.37 0.37 0.37
0.33 0.37 0.39 0.33 0.35 0.37 0.37 0.37 0.37
0.37 0.37 0.33 0.35 0.37 0.37 0.37 0.37 0.37
0.37 0.37 0.37 0.37 0.37 0.37 0.37 0.37 0.37

Layer 1 image number 1 feature number 7
0.27 0.31 0.31 0.27 0.27 0.35 0.39 0.39 0.39
0.29 0.39 0.27 0.29 0.31 0.35 0.39 0.39 0.39
0.35 0.39 0.29 0.31 0.33 0.35 0.39 0.39 0.39
0.39 0.31 0.31 0.33 0.35 0.39 0.39 0.39 0.39
0.37 0.29 0.33 0.35 0.37 0.42 0.39 0.39 0.39
0.39 0.35 0.35 0.37 0.39 0.39 0.39 0.39 0.39
0.39 0.39 0.37 0.39 0.42 0.39 0.39 0.39 0.39
0.44 0.44 0.44 0.42 0.39 0.39 0.39 0.39 0.39
0.39 0.39 0.39 0.39 0.39 0.39 0.39 0.39 0.39

Layer 1 image number 1 feature number 8
0.31 0.35 0.44 0.51 0.59 1.00 1.00 1.00 1.00
0.37 0.39 0.51 0.55 0.59 1.00 1.00 1.00 1.00
0.44 0.44 0.55 0.59 0.65 1.00 1.00 1.00 1.00
0.51 0.51 0.59 0.65 0.73 1.00 1.00 1.00 1.00
0.55 0.55 0.65 0.73 0.86 1.00 1.00 1.00 1.00
0.59 0.59 0.73 0.86 1.00 1.00 1.00 1.00 1.00
0.73 0.73 0.86 1.00 1.00 1.00 1.00 1.00 1.00
1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00
1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00

Layer 1 image number 1 feature number 9
0.27 0.24 0.27 0.24 0.27 0.31 0.27 0.27 0.27
0.26 0.24 0.21 0.23 0.27 0.31 0.27 0.27 0.27
0.27 0.27 0.23 0.24 0.29 0.31 0.27 0.27 0.27
0.27 0.31 0.27 0.26 0.31 0.31 0.27 0.27 0.27
0.20 0.26 0.26 0.27 0.33 0.29 0.27 0.27 0.27
0.24 0.27 0.27 0.29 0.31 0.27 0.27 0.27 0.27
0.27 0.27 0.29 0.31 0.29 0.27 0.27 0.27 0.27
0.31 0.31 0.31 0.29 0.27 0.27 0.27 0.27 0.27
0.27 0.27 0.27 0.27 0.27 0.27 0.27 0.27 0.27

Layer 1 image number 1 feature number 10
0.23 0.23 0.20 0.20 0.26 0.29 0.29 0.29 0.29
0.24 0.29 0.20 0.24 0.29 0.26 0.29 0.29 0.29
0.33 0.29 0.27 0.29 0.24 0.26 0.29 0.29 0.29
0.26 0.23 0.29 0.27 0.23 0.26 0.29 0.29 0.29
0.24 0.27 0.27 0.23 0.24 0.27 0.29 0.29 0.29
0.29 0.26 0.23 0.24 0.26 0.29 0.29 0.29 0.29
0.29 0.23 0.24 0.26 0.27 0.29 0.29 0.29 0.29
0.26 0.26 0.26 0.27 0.29 0.29 0.29 0.29 0.29
0.29 0.29 0.29 0.29 0.29 0.29 0.29 0.29 0.29

```
Layer 1 image number 1 feature number 11
 0.29  0.33  0.20  0.17  0.33  0.26  0.26  0.26  0.26
 0.21  0.23  0.23  0.24  0.29  0.26  0.26  0.26  0.26
 0.26  0.26  0.24  0.26  0.21  0.29  0.26  0.26  0.26
 0.23  0.23  0.26  0.27  0.26  0.26  0.26  0.26  0.26
 0.27  0.27  0.24  0.23  0.31  0.24  0.26  0.26  0.26
 0.23  0.23  0.26  0.27  0.26  0.26  0.26  0.26  0.26
 0.23  0.23  0.27  0.29  0.24  0.26  0.26  0.26  0.26
 0.29  0.29  0.26  0.24  0.26  0.26  0.26  0.26  0.26
 0.26  0.26  0.26  0.26  0.26  0.26  0.26  0.26  0.26
```

Results A3.1 Results for image 1

```
Layer 1 image number 32
  3   1   8   8   8   8  10   8   6
  2   7   8   8   8   8   7   7   6
  8   3   8   8   8   8  10   8   6
  7   8   8   8   8   8   7   8   6
  8   8   8   8   8   8   7   8   2
  8   8   8   8   8   8   8   8   5
  8   8   8   8   8   8   8   8  10
  8   8   8   8   8   8   8   7   9
  1   8   8   8   8   8   8   8   4

Layer 1 image number 32
1.000 0.368 0.444 0.508 0.548 0.508 0.444 0.391 1.000
0.307 0.326 0.444 0.508 0.548 0.474 0.444 0.346 0.508
0.326 0.346 0.444 0.474 0.474 0.474 0.474 0.346 0.444
0.346 0.368 0.444 0.474 0.474 0.474 0.391 0.368 0.346
0.346 0.368 0.444 0.474 0.508 0.508 0.368 0.307 0.368
0.346 0.368 0.474 0.548 0.594 0.594 0.416 0.326 0.346
0.368 0.416 0.508 0.594 0.652 0.594 0.444 0.346 0.346
0.391 0.474 0.548 0.729 0.652 0.548 0.444 0.346 0.391
1.000 0.416 0.508 0.652 0.652 0.594 0.474 0.368 1.000

Layer 1 image number 32 feature number 1
 0.23  0.37  0.26  0.26  0.39  0.29  0.27  0.33  0.35
 0.26  0.27  0.33  0.37  0.27  0.21  0.26  0.29  0.31
 0.24  0.23  0.37  0.35  0.24  0.24  0.20  0.20  0.24
 0.20  0.27  0.33  0.27  0.24  0.21  0.17  0.21  0.24
 0.26  0.27  0.26  0.24  0.29  0.23  0.24  0.26  0.27
 0.26  0.24  0.24  0.31  0.26  0.26  0.21  0.21  0.19
 0.21  0.21  0.37  0.29  0.27  0.26  0.26  0.20  0.21
 0.26  0.44  0.27  0.29  0.27  0.35  0.29  0.26  0.23
 1.00  0.35  0.33  0.31  0.35  0.26  0.27  0.31  0.42

Layer 1 image number 32 feature number 2
 0.27  0.31  0.21  0.19  0.33  0.24  0.26  0.24  0.23
 0.31  0.26  0.24  0.24  0.29  0.26  0.24  0.24  0.29
 0.20  0.24  0.27  0.29  0.23  0.29  0.21  0.21  0.17
 0.24  0.29  0.27  0.23  0.29  0.23  0.21  0.23  0.26
 0.24  0.29  0.24  0.26  0.31  0.16  0.17  0.19  0.37
 0.19  0.20  0.29  0.33  0.24  0.24  0.20  0.15  0.15
 0.20  0.23  0.31  0.31  0.23  0.21  0.19  0.14  0.15
 0.24  0.29  0.26  0.27  0.26  0.26  0.27  0.27  0.31
 0.37  0.29  0.27  0.26  0.26  0.21  0.23  0.23  0.35

Layer 1 image number 32 feature number 3
 1.00  0.16  0.16  0.31  0.20  0.39  0.33  0.21  0.23
 0.31  0.23  0.21  0.24  0.29  0.29  0.31  0.27  0.23
 0.13  0.35  0.27  0.17  0.33  0.33  0.27  0.21  0.20
 0.24  0.23  0.24  0.33  0.29  0.26  0.24  0.23  0.20
 0.27  0.15  0.27  0.42  0.27  0.24  0.26  0.21  0.23
 0.19  0.23  0.29  0.29  0.31  0.31  0.26  0.23  0.23
 0.20  0.26  0.27  0.27  0.29  0.31  0.24  0.31  0.26
 0.27  0.23  0.26  0.31  0.29  0.33  0.31  0.24  0.21
 0.23  0.26  0.27  0.26  0.33  0.24  0.26  0.33  0.31
```

Layer 1 image number 32 feature number 4
0.31 0.31 0.24 0.21 0.29 0.24 0.37 0.27 0.26
0.27 0.23 0.27 0.27 0.29 0.20 0.35 0.27 0.33
0.20 0.27 0.27 0.29 0.20 0.29 0.31 0.24 0.17
0.27 0.20 0.24 0.23 0.29 0.26 0.24 0.23 0.23
0.27 0.23 0.21 0.23 0.27 0.24 0.29 0.27 0.33
0.31 0.23 0.26 0.23 0.31 0.27 0.26 0.23 0.26
0.15 0.23 0.27 0.31 0.26 0.35 0.27 0.24 0.20
0.27 0.29 0.29 0.27 0.26 0.33 0.31 0.27 0.21
0.42 0.33 0.27 0.26 0.33 0.21 0.26 0.29 1.00

Layer 1 image number 32 feature number 5
0.26 0.29 0.23 0.23 0.35 0.26 0.39 0.23 0.21
0.23 0.21 0.23 0.26 0.31 0.27 0.42 0.26 0.21
0.31 0.23 0.23 0.24 0.24 0.31 0.33 0.23 0.21
0.26 0.21 0.23 0.24 0.31 0.27 0.29 0.24 0.24
0.26 0.24 0.23 0.21 0.37 0.33 0.35 0.29 0.27
0.29 0.24 0.27 0.27 0.33 0.33 0.31 0.27 0.35
0.19 0.24 0.33 0.37 0.31 0.37 0.29 0.26 0.27
0.29 0.31 0.31 0.29 0.35 0.31 0.29 0.26 0.33
0.31 0.24 0.23 0.27 0.35 0.29 0.31 0.31 0.55

Layer 1 image number 32 feature number 6
0.23 0.23 0.42 0.37 0.27 0.29 0.27 0.33 1.00
0.26 0.24 0.42 0.37 0.24 0.24 0.26 0.29 0.51
0.24 0.23 0.33 0.31 0.31 0.24 0.23 0.23 0.44
0.23 0.24 0.33 0.31 0.24 0.21 0.23 0.21 0.35
0.26 0.27 0.33 0.27 0.23 0.26 0.27 0.26 0.35
0.29 0.27 0.31 0.27 0.26 0.26 0.24 0.21 0.27
0.27 0.31 0.33 0.26 0.27 0.26 0.26 0.26 0.27
0.33 0.35 0.31 0.29 0.31 0.27 0.29 0.26 0.26
0.35 0.39 0.37 0.39 0.31 0.29 0.35 0.35 0.26

Layer 1 image number 32 feature number 7
0.27 0.31 0.31 0.27 0.26 0.31 0.37 0.35 0.23
0.27 0.33 0.24 0.27 0.29 0.29 0.44 0.35 0.26
0.29 0.35 0.24 0.26 0.26 0.29 0.39 0.27 0.23
0.35 0.26 0.24 0.26 0.26 0.33 0.39 0.26 0.23
0.31 0.23 0.24 0.26 0.31 0.35 0.37 0.24 0.23
0.27 0.26 0.26 0.29 0.35 0.39 0.37 0.26 0.20
0.29 0.26 0.27 0.31 0.42 0.35 0.35 0.27 0.23
0.27 0.29 0.33 0.35 0.37 0.42 0.39 0.35 0.35
0.29 0.33 0.31 0.33 0.33 0.39 0.37 0.33 0.31

Layer 1 image number 32 feature number 8
0.31 0.35 0.44 0.51 0.55 0.51 0.42 0.39 0.37
0.31 0.33 0.44 0.51 0.55 0.47 0.39 0.35 0.33
0.33 0.35 0.44 0.47 0.47 0.47 0.39 0.35 0.33
0.35 0.37 0.44 0.47 0.47 0.47 0.39 0.37 0.33
0.35 0.37 0.44 0.47 0.51 0.51 0.37 0.31 0.29
0.35 0.37 0.47 0.55 0.59 0.59 0.42 0.33 0.29
0.37 0.42 0.51 0.59 0.65 0.59 0.44 0.35 0.29
0.39 0.47 0.55 0.73 0.65 0.55 0.44 0.35 0.27
0.33 0.42 0.51 0.65 0.65 0.59 0.47 0.37 0.31

Layer 1 image number 32 feature number 9
0.27 0.24 0.27 0.24 0.26 0.24 0.26 0.24 0.26
0.31 0.29 0.24 0.24 0.29 0.26 0.24 0.24 0.26
0.23 0.24 0.21 0.23 0.29 0.29 0.21 0.21 0.17
0.24 0.26 0.24 0.23 0.26 0.26 0.21 0.23 0.23
0.24 0.29 0.27 0.23 0.27 0.19 0.17 0.19 0.33
0.19 0.26 0.26 0.29 0.27 0.24 0.20 0.15 0.15
0.23 0.23 0.27 0.31 0.26 0.21 0.19 0.14 0.15
0.24 0.26 0.29 0.31 0.26 0.26 0.27 0.27 0.39
0.33 0.37 0.35 0.33 0.29 0.21 0.23 0.23 0.31

```
Layer 1 image number 32 feature number 10
 0.23  0.23  0.20  0.20  0.27  0.29  0.44  0.20  0.19
 0.20  0.24  0.17  0.23  0.31  0.27  0.37  0.20  0.16
 0.27  0.23  0.23  0.27  0.27  0.27  0.47  0.29  0.27
 0.29  0.24  0.29  0.24  0.24  0.24  0.37  0.24  0.31
 0.20  0.24  0.26  0.24  0.26  0.29  0.31  0.26  0.21
 0.29  0.21  0.21  0.21  0.29  0.29  0.31  0.24  0.24
 0.24  0.19  0.20  0.26  0.31  0.37  0.37  0.33  0.35
 0.20  0.19  0.27  0.23  0.31  0.27  0.29  0.23  0.23
 0.16  0.24  0.17  0.21  0.27  0.33  0.24  0.21  0.23

Layer 1 image number 32 feature number 11
 0.29  0.33  0.20  0.17  0.31  0.23  0.31  0.23  0.21
 0.26  0.24  0.23  0.23  0.31  0.19  0.29  0.23  0.27
 0.19  0.26  0.26  0.27  0.19  0.27  0.26  0.20  0.16
 0.23  0.21  0.26  0.24  0.27  0.21  0.20  0.19  0.21
 0.26  0.24  0.20  0.21  0.29  0.20  0.24  0.23  0.31
 0.29  0.19  0.24  0.24  0.29  0.23  0.21  0.19  0.27
 0.12  0.21  0.29  0.33  0.21  0.29  0.23  0.20  0.16
 0.26  0.27  0.27  0.23  0.24  0.27  0.26  0.23  0.20
 0.39  0.31  0.23  0.24  0.27  0.17  0.21  0.24  0.86
```

Results A3.2 Results for image 32

In Results A3.1, the first rectangle shows the neuron with the maximum output and the second rectangle shows the maximum output value. By comparing these two rectangles and Image 1 in Figure A3.1, it is evident that neuron 3 has learnt and classified the number 0 and neuron 8 has learnt and classified the background. A similar analysis can be carried out for Results A3.2.

Partial results for the other images are given in Results A3.3.

```
Layer 1 image number 2
  6   8   8   8   8   8   8   8   8
  6   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8

Layer 1 image number 2
1.000 0.391 0.444 1.000 1.000 1.000 1.000 1.000 1.000
0.416 0.416 0.474 1.000 1.000 1.000 1.000 1.000 1.000
0.444 0.474 0.508 1.000 1.000 1.000 1.000 1.000 1.000
0.474 0.508 0.548 1.000 1.000 1.000 1.000 1.000 1.000
0.508 0.548 0.594 1.000 1.000 1.000 1.000 1.000 1.000
0.548 0.594 0.652 1.000 1.000 1.000 1.000 1.000 1.000
0.594 0.652 0.729 1.000 1.000 1.000 1.000 1.000 1.000
0.652 0.729 0.863 1.000 1.000 1.000 1.000 1.000 1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
```

```
Layer 1 image number 3
  1   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
```

```
Layer 1 image number 3
1.000 0.416 0.508 0.652 1.000 1.000 1.000 1.000 1.000
0.391 0.508 0.594 0.729 1.000 1.000 1.000 1.000 1.000
0.444 0.548 0.652 0.863 1.000 1.000 1.000 1.000 1.000
0.474 0.594 0.729 1.000 1.000 1.000 1.000 1.000 1.000
0.508 0.652 0.863 1.000 1.000 1.000 1.000 1.000 1.000
0.548 0.729 0.863 1.000 1.000 1.000 1.000 1.000 1.000
0.594 0.729 0.863 1.000 1.000 1.000 1.000 1.000 1.000
0.652 0.729 0.863 1.000 1.000 1.000 1.000 1.000 1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
```

```
Layer 1 image number 4
  4   7   6   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
 10   8   8   8   8   8   8   8   8
  7   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
```

```
Layer 1 image number 4
1.000 0.346 0.444 0.508 0.652 1.000 1.000 1.000 1.000
0.368 0.416 0.474 0.548 0.652 1.000 1.000 1.000 1.000
0.444 0.444 0.508 0.594 0.729 1.000 1.000 1.000 1.000
0.508 0.474 0.548 0.652 0.863 1.000 1.000 1.000 1.000
0.548 0.594 0.652 0.729 0.863 1.000 1.000 1.000 1.000
0.594 0.652 0.729 0.863 1.000 1.000 1.000 1.000 1.000
0.652 0.729 0.863 1.000 1.000 1.000 1.000 1.000 1.000
0.863 0.863 1.000 1.000 1.000 1.000 1.000 1.000 1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
```

```
Layer 1 image number 5
 10   8   6   8   8   8   8   8   8
  3   8   6   8   8   8   8   8   8
  5   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
  7   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
```

```
Layer 1 image number 5
1.000 0.326 0.508 0.416 0.474 1.000 1.000 1.000 1.000
0.346 0.346 0.416 0.444 0.508 1.000 1.000 1.000 1.000
0.474 0.391 0.444 0.508 0.548 1.000 1.000 1.000 1.000
0.416 0.444 0.508 0.548 0.594 1.000 1.000 1.000 1.000
0.474 0.508 0.548 0.594 0.652 1.000 1.000 1.000 1.000
0.652 0.548 0.594 0.652 0.729 1.000 1.000 1.000 1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
```

```
Layer 1 image number 6
  9  9  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  5  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8

Layer 1 image number 6
1.000 0.444 0.474 0.594 0.863 1.000 1.000 1.000 1.000
0.368 0.508 0.594 0.729 1.000 1.000 1.000 1.000 1.000
0.391 0.508 0.594 0.729 1.000 1.000 1.000 1.000 1.000
0.416 0.508 0.594 0.729 1.000 1.000 1.000 1.000 1.000
0.474 0.508 0.594 0.729 1.000 1.000 1.000 1.000 1.000
0.594 0.652 0.729 0.863 1.000 1.000 1.000 1.000 1.000
0.652 0.729 0.863 1.000 1.000 1.000 1.000 1.000 1.000
0.729 0.863 1.000 1.000 1.000 1.000 1.000 1.000 1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000

Layer 1 image number 7
  2  9  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  5  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8

Layer 1 image number 7
1.000 0.444 0.474 0.594 0.863 1.000 1.000 1.000 1.000
0.346 0.508 0.594 0.729 1.000 1.000 1.000 1.000 1.000
0.368 0.508 0.594 0.729 1.000 1.000 1.000 1.000 1.000
0.391 0.508 0.594 0.729 1.000 1.000 1.000 1.000 1.000
0.594 0.508 0.594 0.729 1.000 1.000 1.000 1.000 1.000
0.548 0.652 0.729 0.863 1.000 1.000 1.000 1.000 1.000
0.652 0.729 0.863 1.000 1.000 1.000 1.000 1.000 1.000
0.863 0.863 1.000 1.000 1.000 1.000 1.000 1.000 1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000

Layer 1 image number 8
  7  8  8  8  8  8  8  8  8
  7  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8

Layer 1 image number 8
1.000 0.416 0.444 0.474 0.729 1.000 1.000 1.000 1.000
0.548 0.548 0.548 0.548 0.863 1.000 1.000 1.000 1.000
0.594 0.594 0.594 0.594 1.000 1.000 1.000 1.000 1.000
0.652 0.652 0.652 0.652 1.000 1.000 1.000 1.000 1.000
0.729 0.729 0.729 0.729 1.000 1.000 1.000 1.000 1.000
0.863 0.863 0.863 0.863 1.000 1.000 1.000 1.000 1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
```

```
Layer 1 image number 9
 11  7  6  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
 10  8  8  8  8  8  8  8  8
  7  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8

Layer 1 image number 9
1.000 0.346 0.444 0.508 0.652 1.000 1.000 1.000 1.000
0.307 0.416 0.474 0.548 0.652 1.000 1.000 1.000 1.000
0.368 0.444 0.508 0.594 0.729 1.000 1.000 1.000 1.000
0.444 0.474 0.548 0.652 0.863 1.000 1.000 1.000 1.000
0.474 0.594 0.652 0.729 0.863 1.000 1.000 1.000 1.000
0.548 0.652 0.729 0.863 1.000 1.000 1.000 1.000 1.000
0.652 0.729 0.863 1.000 1.000 1.000 1.000 1.000 1.000
0.863 0.863 1.000 1.000 1.000 1.000 1.000 1.000 1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000

Layer 1 image number 10
  5  7  6  8  8  8  8  8  8
  8  8  6  8  8  8  8  8  8
  8  8  6  8  8  8  8  8  8
  7  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8

Layer 1 image number 10
1.000 0.416 0.444 0.474 0.548 1.000 1.000 1.000 1.000
0.391 0.444 0.508 0.508 0.548 1.000 1.000 1.000 1.000
0.444 0.474 0.548 0.548 0.594 1.000 1.000 1.000 1.000
0.594 0.508 0.548 0.594 0.652 1.000 1.000 1.000 1.000
0.729 0.729 0.729 0.729 0.729 1.000 1.000 1.000 1.000
0.863 0.863 0.863 0.863 0.863 1.000 1.000 1.000 1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000

Layer 1 image number 11
  3  1  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8

Layer 1 image number 11
1.000 0.368 0.416 0.474 0.594 1.000 1.000 1.000 1.000
0.368 0.391 0.474 0.508 0.594 1.000 1.000 1.000 1.000
0.416 0.416 0.508 0.548 0.652 1.000 1.000 1.000 1.000
0.474 0.474 0.548 0.594 0.729 1.000 1.000 1.000 1.000
0.548 0.548 0.652 0.729 0.863 1.000 1.000 1.000 1.000
0.594 0.594 0.729 0.863 1.000 1.000 1.000 1.000 1.000
0.729 0.729 0.863 1.000 1.000 1.000 1.000 1.000 1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
```

```
Layer 1 image number 12
  6  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8

Layer 1 image number 12
1.000 0.391 0.444 0.863 1.000 1.000 1.000 1.000 1.000
0.391 0.416 0.474 0.863 1.000 1.000 1.000 1.000 1.000
0.444 0.474 0.508 0.863 1.000 1.000 1.000 1.000 1.000
0.508 0.548 0.594 1.000 1.000 1.000 1.000 1.000 1.000
0.548 0.594 0.652 1.000 1.000 1.000 1.000 1.000 1.000
0.594 0.652 0.729 1.000 1.000 1.000 1.000 1.000 1.000
0.594 0.652 0.729 1.000 1.000 1.000 1.000 1.000 1.000
0.652 0.729 0.863 1.000 1.000 1.000 1.000 1.000 1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000

Layer 1 image number 13
  1  6  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  5  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8

Layer 1 image number 13
1.000 0.444 0.508 0.594 1.000 1.000 1.000 1.000 1.000
0.391 0.508 0.594 0.652 1.000 1.000 1.000 1.000 1.000
0.444 0.548 0.652 0.729 1.000 1.000 1.000 1.000 1.000
0.474 0.594 0.729 0.863 1.000 1.000 1.000 1.000 1.000
0.508 0.594 0.729 0.863 1.000 1.000 1.000 1.000 1.000
0.548 0.729 0.863 1.000 1.000 1.000 1.000 1.000 1.000
0.594 0.729 0.863 1.000 1.000 1.000 1.000 1.000 1.000
0.652 0.729 0.863 1.000 1.000 1.000 1.000 1.000 1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000

Layer 1 image number 14
  4  7  6  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  7  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8

Layer 1 image number 14
1.000 0.346 0.444 0.548 0.652 1.000 1.000 1.000 1.000
0.368 0.416 0.474 0.594 0.652 1.000 1.000 1.000 1.000
0.416 0.444 0.508 0.652 0.729 1.000 1.000 1.000 1.000
0.444 0.474 0.548 0.729 0.863 1.000 1.000 1.000 1.000
0.508 0.548 0.594 0.729 0.863 1.000 1.000 1.000 1.000
0.548 0.594 0.652 0.863 1.000 1.000 1.000 1.000 1.000
0.652 0.729 0.863 1.000 1.000 1.000 1.000 1.000 1.000
0.863 0.863 1.000 1.000 1.000 1.000 1.000 1.000 1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
```

```
Layer 1 image number 15
 10  8  6  8  8  8  8  8  8
  3  7  6  8  8  8  8  8  8
  5  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  7  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8

Layer 1 image number 15
1.000 0.326 0.548 0.416 0.474 1.000 1.000 1.000 1.000
0.346 0.346 0.444 0.444 0.508 1.000 1.000 1.000 1.000
0.444 0.416 0.474 0.508 0.548 1.000 1.000 1.000 1.000
0.444 0.474 0.548 0.548 0.594 1.000 1.000 1.000 1.000
0.508 0.548 0.594 0.594 0.652 1.000 1.000 1.000 1.000
0.594 0.594 0.652 0.652 0.729 1.000 1.000 1.000 1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000

Layer 1 image number 16
  9  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  5  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8

Layer 1 image number 16
1.000 0.391 0.444 0.594 0.863 1.000 1.000 1.000 1.000
0.346 0.474 0.548 0.729 1.000 1.000 1.000 1.000 1.000
0.368 0.474 0.548 0.729 1.000 1.000 1.000 1.000 1.000
0.416 0.508 0.594 0.729 1.000 1.000 1.000 1.000 1.000
0.474 0.508 0.594 0.729 1.000 1.000 1.000 1.000 1.000
0.594 0.652 0.729 0.863 1.000 1.000 1.000 1.000 1.000
0.652 0.729 0.863 1.000 1.000 1.000 1.000 1.000 1.000
0.729 0.863 1.000 1.000 1.000 1.000 1.000 1.000 1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000

Layer 1 image number 17
  2  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  5  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8

Layer 1 image number 17
1.000 0.391 0.474 0.652 0.863 1.000 1.000 1.000 1.000
0.326 0.474 0.548 0.729 1.000 1.000 1.000 1.000 1.000
0.346 0.474 0.548 0.729 1.000 1.000 1.000 1.000 1.000
0.368 0.474 0.548 0.729 1.000 1.000 1.000 1.000 1.000
0.548 0.474 0.548 0.729 1.000 1.000 1.000 1.000 1.000
0.508 0.594 0.652 0.863 1.000 1.000 1.000 1.000 1.000
0.652 0.729 0.863 1.000 1.000 1.000 1.000 1.000 1.000
0.863 0.863 1.000 1.000 1.000 1.000 1.000 1.000 1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
```

```
Layer 1 image number 18
  7  8  8  8  8  8  8  8  8
  7  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8

Layer 1 image number 18
1.000 0.416 0.474 0.508 0.729 1.000 1.000 1.000 1.000
0.508 0.508 0.548 0.548 0.863 1.000 1.000 1.000 1.000
0.548 0.548 0.594 0.594 1.000 1.000 1.000 1.000 1.000
0.594 0.594 0.652 0.652 1.000 1.000 1.000 1.000 1.000
0.729 0.729 0.729 0.729 1.000 1.000 1.000 1.000 1.000
0.863 0.863 0.863 0.863 1.000 1.000 1.000 1.000 1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000

Layer 1 image number 19
 11  8  6  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
 10  8  8  8  8  8  8  8  8
  7  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8

Layer 1 image number 19
1.000 0.346 0.416 0.508 0.652 1.000 1.000 1.000 1.000
0.289 0.391 0.444 0.548 0.652 1.000 1.000 1.000 1.000
0.346 0.416 0.474 0.594 0.729 1.000 1.000 1.000 1.000
0.416 0.444 0.508 0.652 0.863 1.000 1.000 1.000 1.000
0.444 0.548 0.594 0.729 0.863 1.000 1.000 1.000 1.000
0.508 0.594 0.652 0.863 1.000 1.000 1.000 1.000 1.000
0.652 0.729 0.863 1.000 1.000 1.000 1.000 1.000 1.000
0.863 0.863 1.000 1.000 1.000 1.000 1.000 1.000 1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000

Layer 1 image number 20
  5  7  6  8  8  8  8  8  8
  7  8  6  8  8  8  8  8  8
  7  8  6  8  8  8  8  8  8
  7  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8

Layer 1 image number 20
1.000 0.368 0.508 0.474 0.548 1.000 1.000 1.000 1.000
0.368 0.416 0.474 0.474 0.548 1.000 1.000 1.000 1.000
0.416 0.444 0.508 0.508 0.594 1.000 1.000 1.000 1.000
0.652 0.474 0.508 0.548 0.652 1.000 1.000 1.000 1.000
0.652 0.652 0.652 0.652 0.729 1.000 1.000 1.000 1.000
0.729 0.729 0.729 0.729 0.863 1.000 1.000 1.000 1.000
0.863 0.863 0.863 0.863 1.000 1.000 1.000 1.000 1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
```

```
Layer 1 image number 21
  8   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
 10   8   8   8   8   8   8   8   8
  3   1   8   8   8   8   8   8   8

Layer 1 image number 21
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
0.729 0.729 0.863 1.000 1.000 1.000 1.000 1.000 1.000
0.594 0.594 0.729 0.863 1.000 1.000 1.000 1.000 1.000
0.548 0.548 0.652 0.729 0.863 1.000 1.000 1.000 1.000
0.508 0.508 0.594 0.652 0.729 1.000 1.000 1.000 1.000
0.416 0.444 0.548 0.594 0.652 1.000 1.000 1.000 1.000
0.368 0.391 0.508 0.548 0.594 1.000 1.000 1.000 1.000
1.000 0.368 0.444 0.508 0.594 1.000 1.000 1.000 1.000

Layer 1 image number 22
  8   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
  6   8   8   8   8   8   8   8   8
  6   8   8   8   8   8   8   8   8

Layer 1 image number 22
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
0.863 0.863 1.000 1.000 1.000 1.000 1.000 1.000 1.000
0.729 0.729 0.863 1.000 1.000 1.000 1.000 1.000 1.000
0.652 0.652 0.729 1.000 1.000 1.000 1.000 1.000 1.000
0.594 0.594 0.652 1.000 1.000 1.000 1.000 1.000 1.000
0.548 0.548 0.594 1.000 1.000 1.000 1.000 1.000 1.000
0.548 0.508 0.548 1.000 1.000 1.000 1.000 1.000 1.000
1.000 0.391 0.444 1.000 1.000 1.000 1.000 1.000 1.000

Layer 1 image number 23
  8   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
  1   8   8   8   8   8   8   8   8

Layer 1 image number 23
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
0.863 0.863 1.000 1.000 1.000 1.000 1.000 1.000 1.000
0.652 0.729 0.863 1.000 1.000 1.000 1.000 1.000 1.000
0.594 0.652 0.729 0.863 1.000 1.000 1.000 1.000 1.000
0.548 0.594 0.652 0.863 1.000 1.000 1.000 1.000 1.000
0.508 0.548 0.652 0.863 1.000 1.000 1.000 1.000 1.000
0.474 0.548 0.652 0.863 1.000 1.000 1.000 1.000 1.000
0.444 0.548 0.652 0.863 1.000 1.000 1.000 1.000 1.000
1.000 0.416 0.508 0.652 1.000 1.000 1.000 1.000 1.000
```

```
Layer 1 image number 24
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  2  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  4  7  6  8  8  8  8  8  8

Layer 1 image number 24
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
0.863 0.863 1.000 1.000 1.000 1.000 1.000 1.000 1.000
0.652 0.729 0.863 1.000 1.000 1.000 1.000 1.000 1.000
0.594 0.652 0.729 0.863 1.000 1.000 1.000 1.000 1.000
0.474 0.508 0.594 0.729 1.000 1.000 1.000 1.000 1.000
0.444 0.474 0.548 0.652 0.863 1.000 1.000 1.000 1.000
0.416 0.444 0.508 0.594 0.729 1.000 1.000 1.000 1.000
0.368 0.416 0.474 0.548 0.652 1.000 1.000 1.000 1.000
1.000 0.346 0.444 0.508 0.652 1.000 1.000 1.000 1.000

Layer 1 image number 25
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  4  1  6  8  8  8  8  8  8
  5  8  6  8  8  8  8  8  8
 10  8  6  8  8  8  8  8  8

Layer 1 image number 25
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
0.863 0.863 0.863 0.863 1.000 1.000 1.000 1.000 1.000
0.652 0.652 0.652 0.729 0.863 1.000 1.000 1.000 1.000
0.548 0.548 0.594 0.652 0.729 1.000 1.000 1.000 1.000
0.474 0.508 0.548 0.594 0.652 1.000 1.000 1.000 1.000
0.391 0.444 0.594 0.474 0.548 1.000 1.000 1.000 1.000
0.346 0.346 0.416 0.444 0.508 1.000 1.000 1.000 1.000
1.000 0.326 0.508 0.416 0.474 1.000 1.000 1.000 1.000

Layer 1 image number 26
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
  8  8  8  8  8  8  8  8  8
 10  8  8  8  8  8  8  8  8
  9  9  8  8  8  8  8  8  8

Layer 1 image number 26
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
0.652 0.729 0.863 1.000 1.000 1.000 1.000 1.000 1.000
0.594 0.729 0.863 1.000 1.000 1.000 1.000 1.000 1.000
0.548 0.729 0.863 1.000 1.000 1.000 1.000 1.000 1.000
0.508 0.729 0.863 1.000 1.000 1.000 1.000 1.000 1.000
0.391 0.548 0.652 0.863 1.000 1.000 1.000 1.000 1.000
0.368 0.508 0.594 0.729 1.000 1.000 1.000 1.000 1.000
0.368 0.474 0.548 0.652 0.863 1.000 1.000 1.000 1.000
1.000 0.444 0.474 0.594 0.863 1.000 1.000 1.000 1.000
```

```
Layer 1 image number 26
  8   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
 10   8   8   8   8   8   8   8   8
  9   9   8   8   8   8   8   8   8

Layer 1 image number 26
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
0.652 0.729 0.863 1.000 1.000 1.000 1.000 1.000 1.000
0.594 0.729 0.863 1.000 1.000 1.000 1.000 1.000 1.000
0.548 0.729 0.863 1.000 1.000 1.000 1.000 1.000 1.000
0.508 0.729 0.863 1.000 1.000 1.000 1.000 1.000 1.000
0.391 0.548 0.652 0.863 1.000 1.000 1.000 1.000 1.000
0.368 0.508 0.594 0.729 1.000 1.000 1.000 1.000 1.000
0.368 0.474 0.548 0.652 0.863 1.000 1.000 1.000 1.000
1.000 0.444 0.474 0.594 0.863 1.000 1.000 1.000 1.000

Layer 1 image number 27
  8   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
 10   8   8   8   8   8   8   8   8
  2   9   8   8   8   8   8   8   8

Layer 1 image number 27
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
0.729 0.729 0.863 1.000 1.000 1.000 1.000 1.000 1.000
0.652 0.729 0.863 1.000 1.000 1.000 1.000 1.000 1.000
0.594 0.729 0.863 1.000 1.000 1.000 1.000 1.000 1.000
0.548 0.729 0.863 1.000 1.000 1.000 1.000 1.000 1.000
0.416 0.548 0.652 0.863 1.000 1.000 1.000 1.000 1.000
0.368 0.508 0.594 0.729 1.000 1.000 1.000 1.000 1.000
0.346 0.474 0.548 0.652 0.863 1.000 1.000 1.000 1.000
1.000 0.444 0.474 0.594 0.863 1.000 1.000 1.000 1.000

Layer 1 image number 28
  8   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
  7   8   8   8   8   8   8   8   8
  7   8   8   8   8   8   8   8   8

Layer 1 image number 28
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
0.652 0.729 0.863 1.000 1.000 1.000 1.000 1.000 1.000
0.594 0.652 0.729 0.863 1.000 1.000 1.000 1.000 1.000
0.548 0.594 0.652 0.729 0.863 1.000 1.000 1.000 1.000
0.508 0.548 0.594 0.652 0.729 1.000 1.000 1.000 1.000
0.474 0.508 0.548 0.594 0.729 1.000 1.000 1.000 1.000
0.444 0.474 0.508 0.548 0.729 1.000 1.000 1.000 1.000
0.416 0.444 0.474 0.508 0.729 1.000 1.000 1.000 1.000
1.000 0.416 0.444 0.474 0.729 1.000 1.000 1.000 1.000
```

```
Layer 1 image number 29
  8   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
  2   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
 10   8   8   8   8   8   8   8   8
  2   8   8   8   8   8   8   8   8
 11   7   6   8   8   8   8   8   8

Layer 1 image number 29
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
0.863 0.863 1.000 1.000 1.000 1.000 1.000 1.000 1.000
0.652 0.729 0.863 1.000 1.000 1.000 1.000 1.000 1.000
0.548 0.652 0.729 0.863 1.000 1.000 1.000 1.000 1.000
0.508 0.508 0.594 0.729 1.000 1.000 1.000 1.000 1.000
0.391 0.474 0.548 0.652 0.863 1.000 1.000 1.000 1.000
0.368 0.444 0.508 0.594 0.729 1.000 1.000 1.000 1.000
0.307 0.416 0.474 0.548 0.652 1.000 1.000 1.000 1.000
1.000 0.346 0.444 0.508 0.652 1.000 1.000 1.000 1.000

Layer 1 image number 30
  8   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
  2   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
 10   8   8   8   8   8   8   8   8
  8   8   6   8   8   8   8   8   8
  5   7   6   8   8   8   8   8   8

Layer 1 image number 30
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
0.863 0.863 1.000 1.000 1.000 1.000 1.000 1.000 1.000
0.652 0.729 0.863 1.000 1.000 1.000 1.000 1.000 1.000
0.548 0.652 0.729 0.863 1.000 1.000 1.000 1.000 1.000
0.474 0.474 0.548 0.652 0.863 1.000 1.000 1.000 1.000
0.391 0.444 0.508 0.594 0.729 1.000 1.000 1.000 1.000
0.444 0.416 0.474 0.548 0.652 1.000 1.000 1.000 1.000
0.346 0.391 0.474 0.508 0.594 1.000 1.000 1.000 1.000
1.000 0.416 0.444 0.474 0.548 1.000 1.000 1.000 1.000

Layer 1 image number 31
  8   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8
  8   8   8   8   8   8   8   8   8

Layer 1 image number 31
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
```

Results A3.3 Partial results for images 2 to 31

By examination of Results A3.1, Results A3.2 and Results A3.3 the feature that each feature neuron learns can be determined and the results are given in Table A3.3.

Feature Neuron	Learnt Feature
1	Number 2
2	Number 6
3	Number 0
4	Number 3
5	Number 9
6	Number 1
7	Number 7
8	Background
9	Number 5
10	Number 4
11	Number 8

Table A3.3 Features classified by the neurons

The classification performance can be displayed visually by producing a surface plot showing the confidence level for all the neuron output values. Figure A3.2 shows the surface plot for the maximum neuron output when feature 1 is presented. This shows that that two types of features are classified, the number and the background. The region between these two classified regions has a low value indicating that the neurons in this region have not classified any feature. Inspection of the region surrounding the classified number shows that there is very good discrimination between the classified feature and the non-classified region.

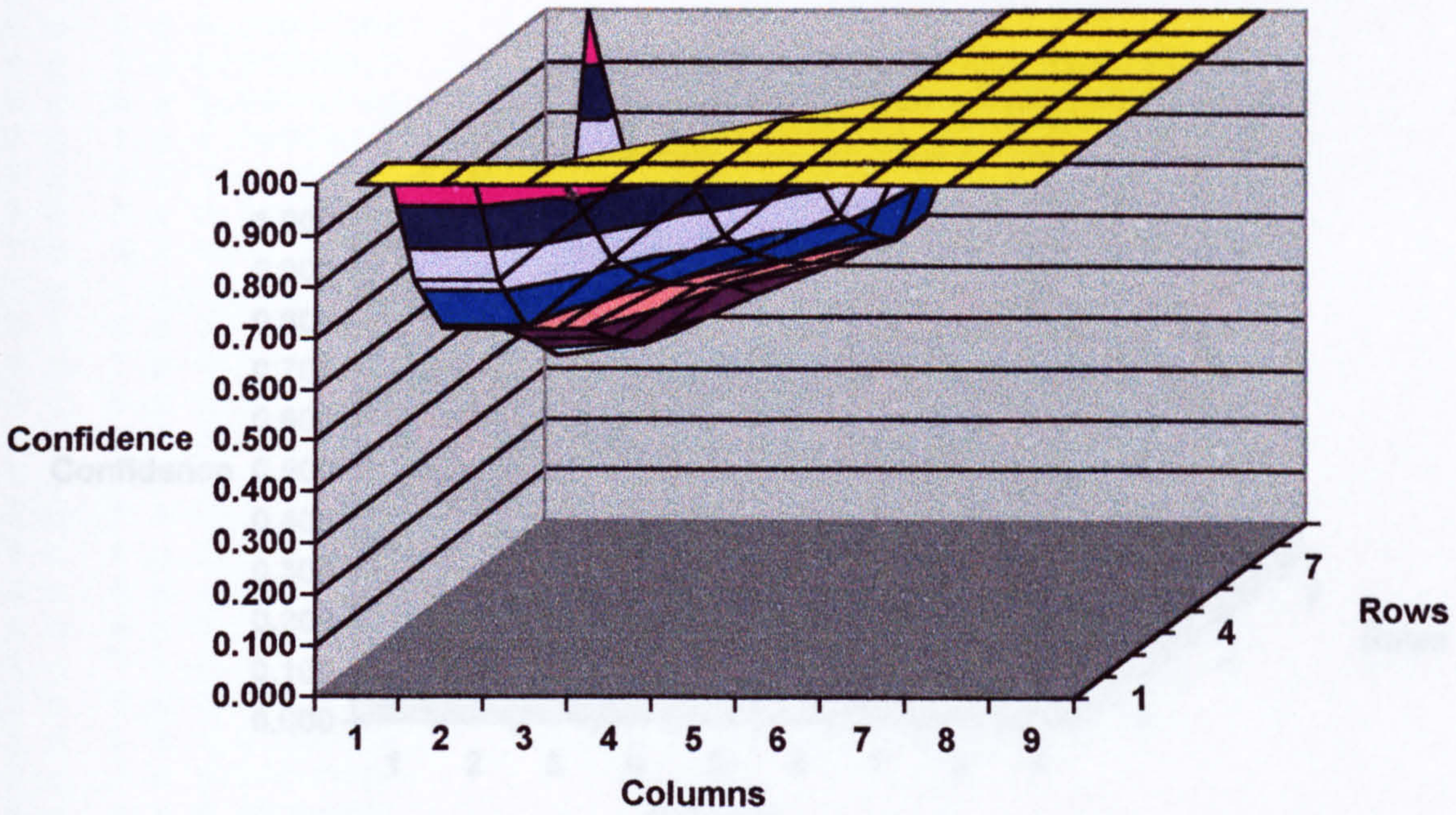


Figure A3.2 Confidence level for image 1

In addition to the network classifying the training set, the Images 11 to 20 show noisy images. Comparing the outputs from presenting Image 1 and 11, Images 2 and 12 to Images 10 and 20 show that the network had successfully classified the noisy images with no performance degradation.

Similarly when comparing the training set classification with the classification of the translated images it is clear that that the images have been successfully classified.

The classification of multiple features in an image is tested by the classification of Image 32, the results of which are shown in Figure A3.3.

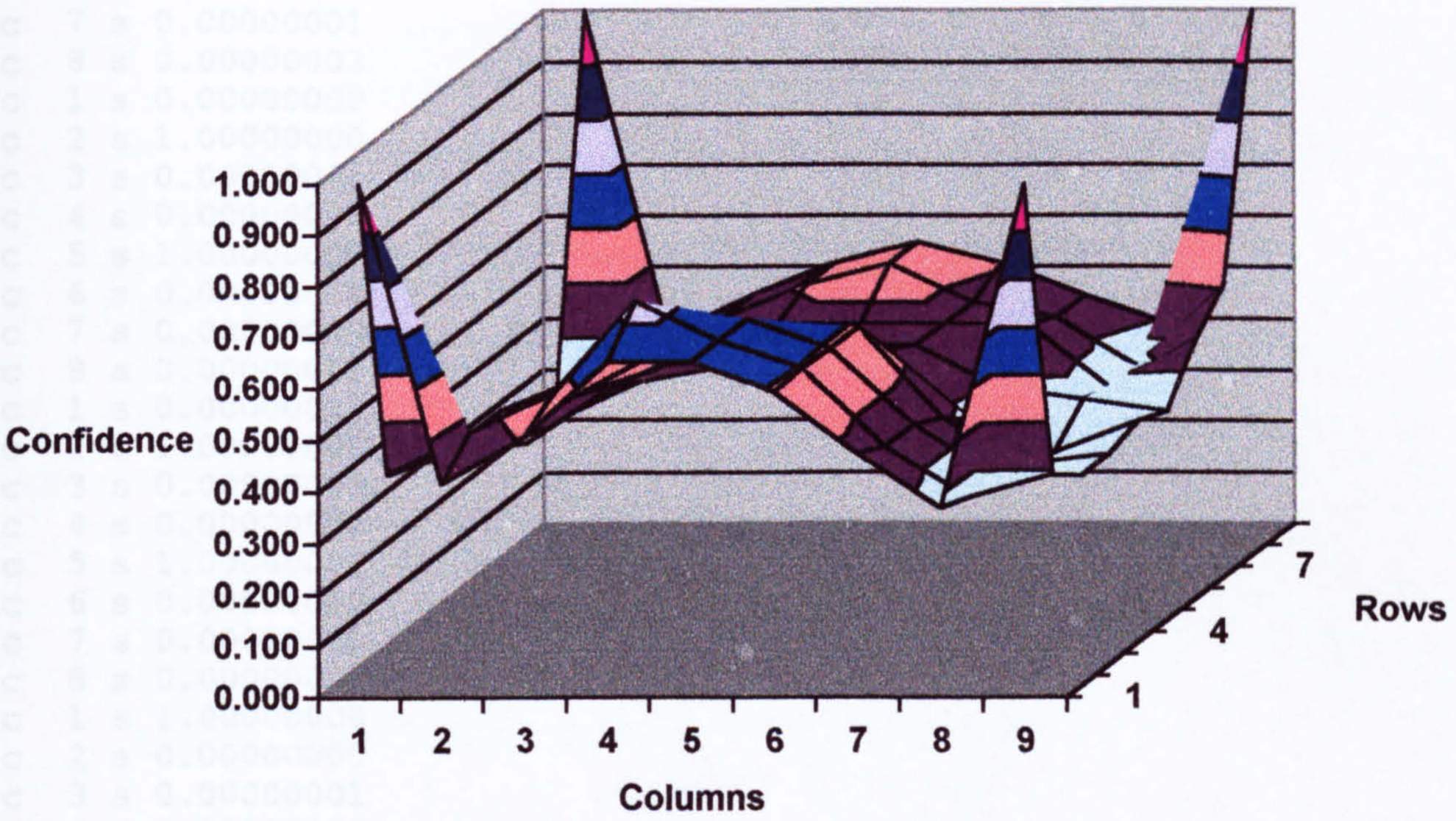


Figure A3.3 Confidence level for image 32

Figure A3.3 shows the confidence level for the classification of a multi-feature image. The four peaks at the locations of the features are clearly identifiable. The *hump* in the middle of the plot is where the background starts to exert an influence. If the size of the retina were increased and the features were further apart, the background would be classified at the centre of the image.

The synapse states have meaningful values for this type of neuron. Table A3.3 shows that Image 1 (Number 0) is classified by feature neuron 3 and has weight values that are shown in Results A3.4.

```
Feature 3
r  8 c  1 s 0.00000002
r  8 c  2 s 0.00000001
r  8 c  3 s 1.00000000
r  8 c  4 s 1.00000000
r  8 c  5 s 0.00000001
r  8 c  6 s 0.00000000
r  8 c  7 s 0.00000001
r  8 c  8 s 0.00000003
r  7 c  1 s 0.00000000
r  7 c  2 s 1.00000000
r  7 c  3 s 0.00000001
r  7 c  4 s 0.00000000
r  7 c  5 s 1.00000000
r  7 c  6 s 0.00000000
r  7 c  7 s 0.00000000
r  7 c  8 s 0.00000001
r  6 c  1 s 0.00000001
r  6 c  2 s 1.00000000
r  6 c  3 s 0.00000003
r  6 c  4 s 0.00000002
r  6 c  5 s 1.00000000
r  6 c  6 s 0.00000000
r  6 c  7 s 0.00000000.
r  6 c  8 s 0.00000002
r  5 c  1 s 1.00000000
r  5 c  2 s 0.00000000
r  5 c  3 s 0.00000001
r  5 c  4 s 0.00000002
r  5 c  5 s 0.00000000
r  5 c  6 s 1.00000000
r  5 c  7 s 0.00000003
r  5 c  8 s 0.00000002
r  4 c  1 s 1.00000000
r  4 c  2 s 0.00000003
r  4 c  3 s 0.00000001
r  4 c  4 s 0.00000003
r  4 c  5 s 0.00000002
r  4 c  6 s 1.00000000
r  4 c  7 s 0.00000003
r  4 c  8 s 0.00000001
r  3 c  1 s 0.00000002
r  3 c  2 s 1.00000000
r  3 c  3 s 0.00000000
r  3 c  4 s 0.00000002
r  3 c  5 s 1.00000000.
r  3 c  6 s 0.00000002
r  3 c  7 s 0.00000003
r  3 c  8 s 0.00000001
r  2 c  1 s 0.00000001
r  2 c  2 s 1.00000000
r  2 c  3 s 0.00000001
r  2 c  4 s 0.00000002
r  2 c  5 s 1.00000000
r  2 c  6 s 0.00000001
r  2 c  7 s 0.00000001
r  2 c  8 s 0.00000002
r  1 c  1 s 0.00000002
r  1 c  2 s 0.00000002
r  1 c  3 s 1.00000000
r  1 c  4 s 1.00000000
r  1 c  5 s 0.00000000
r  1 c  6 s 0.00000003
r  1 c  7 s 0.00000002
r  1 c  8 s 0.00000000
```

Results A3.4 Synapse weights for feature neuron 3

These values can be represented graphically as shown in Figure A3.4.

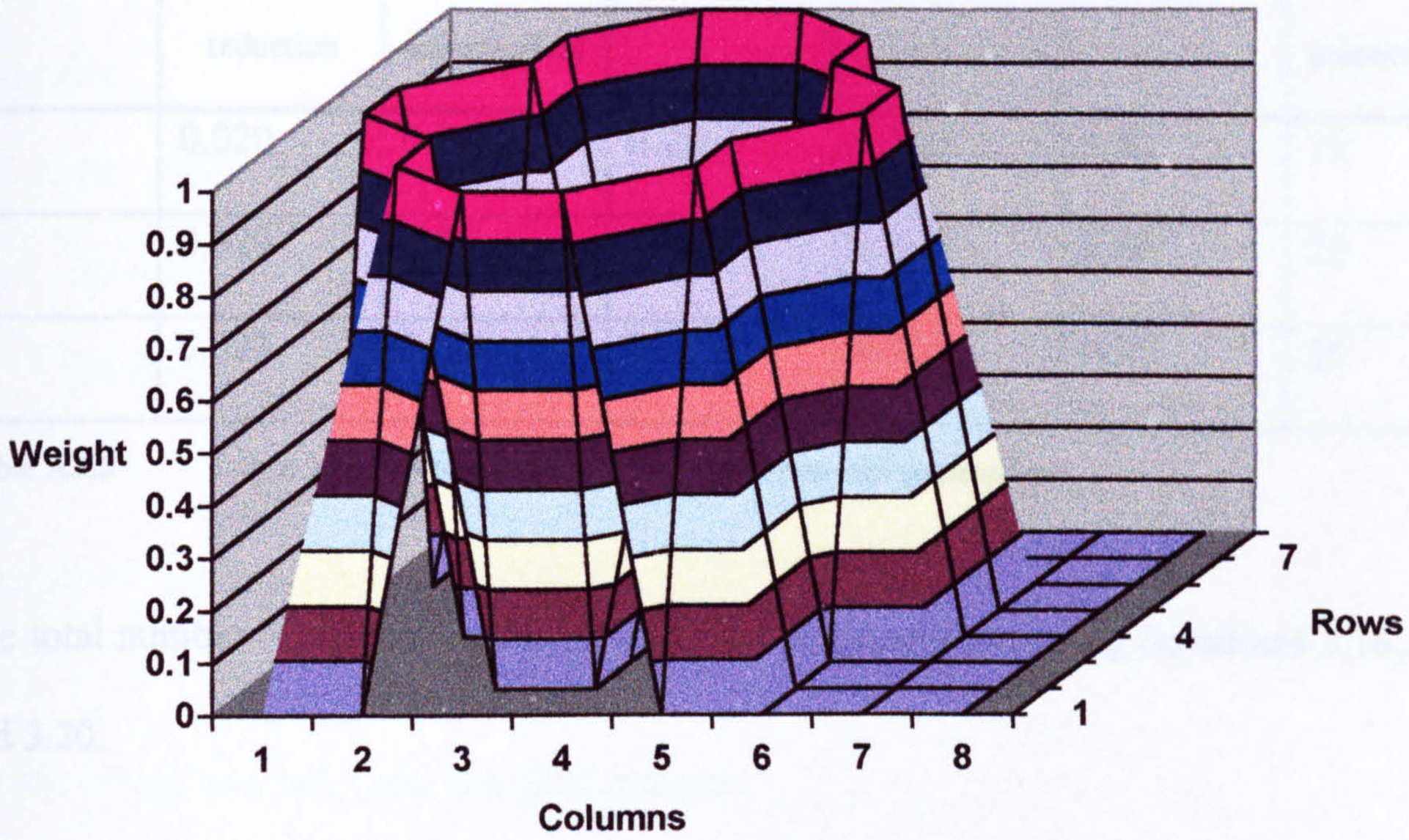


Figure A3.4 Synapse weights for feature number 3

TRIPLE LAYER TESTING

When testing the triple layer network, following characteristics were used:

Layer Number	Number of Features	Receptive Field Rows	Receptive Field Columns
1	16	4	4
2	16	4	64
3	11	2	32

Table A3.4 Triple layer network configuration

The same images as for the single layer network were used. Apart from setting the training session to train all three layers, the teaching parameters were identical to the values used for the single layer network.

The following teaching and neuron run parameters were used:

Layer	Tiredness reduction	Tiredness enhancement	Atrophy	Tolerance	Learning rate	Teaching presentations
1	0.020	0.85	0.25	0.100	0.50	15
2	0.020	0.85	0.25	0.100	0.80	25
3	0.020	0.85	0.25	0.025	0.75	35

Table A3.5 Triple layer network teaching and neuron run parameters

The total number of neurons in the network can be determined using Equations 3.18, 3.19 and 3.20.

For layer 1

There are;

$16 - 4 + 1 = 13$ columns

$16 - 4 + 1 = 13$ rows

16 features

Number of neurons = $13 \times 13 \times 16 = 2704$

For layer 2

There are;

$13 - 64/16 + 1 = 10$ columns

$13 - 4 + 1 = 10$ rows

16 features

Number of neurons = $10 \times 10 \times 16 = 1600$

For layer 3

There are;

$10 - 32/16 + 1 = 9$ columns

$10 - 2 + 1 = 9$ rows

11 features

Number of neurons = $9 \times 9 \times 11 = 924$

For the whole network, there are 5228 neurons.

By introducing two extra layers there are 5.87 as many neurons giving a significant calculation overhead.

When classifying Image 1, the results for the maximum output feature neurons and their associated values for all three layers are shown in Results A3.5.

```
Layer 1 image number 1
 16 12 11 13 12  6  6  6  6  6  6  6  6
 15 12  6 15  9  6  6  6  6  6  6  6  6
 12 16 14  6 16 12  6  6  6  6  6  6  6
  9  6 11  6 15  8  6  6  6  6  6  6  6
 13 12  4 16 12  6  6  6  6  6  6  6  6
 6  5  6  6  8  6  6  6  6  6  6  6  6
 6  9 12 12  6  6  6  6  6  6  6  6  6
 6  6  6  6  6  6  6  6  6  6  6  6  6
 6  6  6  6  6  6  6  6  6  6  6  6  6
 6  6  6  6  6  6  6  6  6  6  6  6  6
 6  6  6  6  6  6  6  6  6  6  6  6  6
 6  6  6  6  6  6  6  6  6  6  6  6  6
```

```
Layer 2 image number 1
  1  2  5  5 15  3  3  3  3  3
  5  5  5  5 15  3  3  3  3  3
  5  5  2 15  3  3  3  3  3  3
  5  5  5 15  3  4  3  3  3  3
  5  2 15 15  3  3  3  3  3  3
  5  5  3  3  4  3  3  3  3  3
  3  3  3  3  3  3  3  3  3  3
  3  3  3  3  3  3  3  3  3  3
  3  3  3  3  3  3  3  3  3  3
  3  3  3  3  3  3  3  3  3  3

Layer 3 image number 1
  8  8  5  5  7  7  7  7  7
  8  8  5  5  7  7  7  7  7
  8  8  5  5  7  7  7  7  7
  8  5  5  5  7  7  7  7  7
  9  5  5  7  7  7  7  7  7
  5  5  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7

Layer 1 image number 1
0.392 0.333 0.286 0.592 0.592 0.592 1.000 1.000 1.000 1.000 1.000 1.000
1.000
0.333 0.333 0.333 0.333 0.469 0.469 1.000 1.000 1.000 1.000 1.000 1.000
1.000
0.469 0.333 0.333 0.333 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
1.000
0.469 0.333 0.469 0.333 0.333 0.469 1.000 1.000 1.000 1.000 1.000 1.000
1.000
1.000 0.333 0.392 0.469 0.592 0.592 1.000 1.000 1.000 1.000 1.000 1.000
1.000
0.333 0.333 0.333 0.392 0.469 1.000 1.000 1.000 1.000 1.000 1.000 1.000
1.000
0.392 0.334 0.392 0.469 0.592 1.000 1.000 1.000 1.000 1.000 1.000 1.000
1.000
0.469 0.469 0.469 0.592 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
1.000

Layer 2 image number 1
0.835 1.000 0.809 0.651 0.609 0.750 1.000 1.000 1.000 1.000
0.799 0.912 0.814 0.656 0.611 0.754 1.000 1.000 1.000 1.000
0.838 1.000 0.681 0.632 0.612 0.805 1.000 1.000 1.000 1.000
0.877 1.000 0.729 0.680 0.667 0.958 1.000 1.000 1.000 1.000
0.808 0.752 0.688 0.625 0.762 1.000 1.000 1.000 1.000 1.000
0.605 0.583 0.618 0.718 0.958 1.000 1.000 1.000 1.000 1.000
0.617 0.649 0.721 0.856 1.000 1.000 1.000 1.000 1.000 1.000
0.770 0.819 0.938 1.000 1.000 1.000 1.000 1.000 1.000 1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
```

```
Layer 3 image number 1
1.000 0.696 0.681 0.634 0.513 0.676 1.000 1.000 1.000
0.792 0.680 0.700 0.619 0.530 0.699 1.000 1.000 1.000
0.749 0.661 0.704 0.576 0.574 0.779 1.000 1.000 1.000
0.735 0.667 0.652 0.504 0.640 1.000 1.000 1.000 1.000
0.648 0.642 0.540 0.580 0.744 1.000 1.000 1.000 1.000
0.572 0.520 0.563 0.692 1.000 1.000 1.000 1.000 1.000
0.552 0.602 0.691 0.829 1.000 1.000 1.000 1.000 1.000
0.711 0.784 0.993 1.000 1.000 1.000 1.000 1.000 1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
```

Results A3.5 Classification of image 1

As the classification results for all three layers would take up many pages, the results for the final layer only are given in Results A3.6.

```
Layer 3 image number 2
 10  5  7  7  7  7  7  7  7
 10  5  7  7  7  7  7  7  7
  5  5  5  7  7  7  7  7  7
  8  5  5  7  7  7  7  7  7
  9  5  5  7  7  7  7  7  7
  5  5  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7

Layer 3 image number 2
1.000 0.494 0.582 0.909 1.000 1.000 1.000 1.000 1.000
0.610 0.524 0.537 0.743 1.000 1.000 1.000 1.000 1.000
0.598 0.604 0.508 0.654 0.891 1.000 1.000 1.000 1.000
0.629 0.668 0.571 0.594 0.783 1.000 1.000 1.000 1.000
0.651 0.648 0.551 0.597 0.786 1.000 1.000 1.000 1.000
0.592 0.556 0.531 0.658 0.903 1.000 1.000 1.000 1.000
0.505 0.544 0.620 0.751 1.000 1.000 1.000 1.000 1.000
0.645 0.681 0.764 0.993 1.000 1.000 1.000 1.000 1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000

Layer 3 image number 3
  5  5  7  7  7  7  7  7  7
  5  5  7  7  7  7  7  7  7
  5  5  5  7  7  7  7  7  7
  8  5  5  7  7  7  7  7  7
  9  5  5  7  7  7  7  7  7
  5  5  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7

Layer 3 image number 3
1.000 0.630 0.517 0.661 1.000 1.000 1.000 1.000 1.000
0.658 0.610 0.521 0.664 1.000 1.000 1.000 1.000 1.000
0.653 0.641 0.509 0.643 0.891 1.000 1.000 1.000 1.000
0.669 0.674 0.560 0.594 0.783 1.000 1.000 1.000 1.000
0.643 0.639 0.546 0.597 0.786 1.000 1.000 1.000 1.000
0.588 0.550 0.537 0.658 0.903 1.000 1.000 1.000 1.000
0.509 0.548 0.623 0.751 1.000 1.000 1.000 1.000 1.000
0.645 0.681 0.764 0.993 1.000 1.000 1.000 1.000 1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
```

```
Layer 3 image number 4
  2  5  5  5  7  7  7  7  7
  8  5  5  5  7  7  7  7  7
  8  5  5  5  7  7  7  7  7
  8  5  5  7  7  7  7  7  7
  9  5  5  7  7  7  7  7  7
  5  5  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7

Layer 3 image number 4
1.000 0.588 0.614 0.510 0.631 1.000 1.000 1.000 1.000
0.664 0.632 0.618 0.510 0.633 1.000 1.000 1.000 1.000
0.729 0.673 0.620 0.500 0.647 1.000 1.000 1.000 1.000
0.731 0.702 0.607 0.526 0.682 1.000 1.000 1.000 1.000
0.655 0.644 0.533 0.591 0.769 1.000 1.000 1.000 1.000
0.579 0.522 0.563 0.692 1.000 1.000 1.000 1.000 1.000
0.538 0.594 0.691 0.829 1.000 1.000 1.000 1.000 1.000
0.686 0.764 0.993 1.000 1.000 1.000 1.000 1.000 1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000

Layer 3 image number 5
  3  9  5  5  7  7  7  7  7
  8  9  5  5  7  7  7  7  7
  8  9  5  5  7  7  7  7  7
  9  9  5  5  7  7  7  7  7
  9  5  5  7  7  7  7  7  7
  5  5  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7

Layer 3 image number 5
0.965 0.799 0.820 0.896 1.000 1.000 1.000 1.000 1.000
0.827 0.799 0.797 0.882 1.000 1.000 1.000 1.000 1.000
0.858 0.829 0.826 0.904 1.000 1.000 1.000 1.000 1.000
0.815 0.827 0.858 0.903 1.000 1.000 1.000 1.000 1.000
0.846 0.823 0.880 0.962 1.000 1.000 1.000 1.000 1.000
0.857 0.867 0.952 1.000 1.000 1.000 1.000 1.000 1.000
0.968 0.952 1.000 1.000 1.000 1.000 1.000 1.000 1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000

Layer 3 image number 6
  1  5  5  7  7  7  7  7  7
  8  5  5  7  7  7  7  7  7
  8  5  5  7  7  7  7  7  7
  9  5  5  7  7  7  7  7  7
  5  5  5  7  7  7  7  7  7
  5  5  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7

Layer 3 image number 6
1.000 0.658 0.584 0.568 0.742 1.000 1.000 1.000 1.000
0.575 0.633 0.608 0.536 0.696 1.000 1.000 1.000 1.000
0.707 0.683 0.630 0.520 0.676 1.000 1.000 1.000 1.000
0.644 0.664 0.607 0.538 0.699 1.000 1.000 1.000 1.000
0.621 0.600 0.531 0.595 0.779 1.000 1.000 1.000 1.000
0.580 0.522 0.563 0.692 1.000 1.000 1.000 1.000 1.000
0.535 0.594 0.691 0.829 1.000 1.000 1.000 1.000 1.000
0.681 0.764 0.993 1.000 1.000 1.000 1.000 1.000 1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
```

```
Layer 3 image number 7
  1  5  5  7  7  7  7  7  7
  8  5  5  7  7  7  7  7  7
  8  5  5  7  7  7  7  7  7
  9  5  5  7  7  7  7  7  7
  5  5  5  7  7  7  7  7  7
  5  5  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7

Layer 3 image number 7
1.000 0.658 0.584 0.568 0.742 1.000 1.000 1.000 1.000
0.579 0.633 0.608 0.536 0.696 1.000 1.000 1.000 1.000
0.704 0.683 0.630 0.520 0.676 1.000 1.000 1.000 1.000
0.643 0.664 0.607 0.538 0.699 1.000 1.000 1.000 1.000
0.621 0.600 0.531 0.595 0.779 1.000 1.000 1.000 1.000
0.579 0.522 0.563 0.692 1.000 1.000 1.000 1.000 1.000
0.538 0.594 0.691 0.829 1.000 1.000 1.000 1.000 1.000
0.686 0.764 0.993 1.000 1.000 1.000 1.000 1.000 1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000

Layer 3 image number 8
  9  5  5  7  7  7  7  7  7
  9  5  5  7  7  7  7  7  7
  5 10  5  7  7  7  7  7  7
  5 10  5  7  7  7  7  7  7
  5  5  7  7  7  7  7  7  7
  5  7  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7

Layer 3 image number 8
1.000 0.666 0.613 0.508 0.675 1.000 1.000 1.000 1.000
0.701 0.643 0.578 0.555 0.769 1.000 1.000 1.000 1.000
0.641 0.602 0.531 0.595 1.000 1.000 1.000 1.000 1.000
0.657 0.578 0.499 0.627 1.000 1.000 1.000 1.000 1.000
0.625 0.548 0.532 0.675 1.000 1.000 1.000 1.000 1.000
0.524 0.533 0.619 0.769 1.000 1.000 1.000 1.000 1.000
0.608 0.657 0.752 1.000 1.000 1.000 1.000 1.000 1.000
0.802 0.905 1.000 1.000 1.000 1.000 1.000 1.000 1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000

Layer 3 image number 9
  2  5  5  5  7  7  7  7  7
  8  5  5  5  7  7  7  7  7
  8  5  5  5  7  7  7  7  7
  8  5  5  7  7  7  7  7  7
  9  5  5  7  7  7  7  7  7
  5  5  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7

Layer 3 image number 9
1.000 0.588 0.614 0.510 0.631 1.000 1.000 1.000 1.000
0.665 0.632 0.618 0.510 0.633 1.000 1.000 1.000 1.000
0.723 0.673 0.620 0.500 0.647 1.000 1.000 1.000 1.000
0.717 0.702 0.607 0.526 0.682 1.000 1.000 1.000 1.000
0.656 0.644 0.533 0.591 0.769 1.000 1.000 1.000 1.000
0.579 0.522 0.563 0.692 1.000 1.000 1.000 1.000 1.000
0.538 0.594 0.691 0.829 1.000 1.000 1.000 1.000 1.000
0.686 0.764 0.993 1.000 1.000 1.000 1.000 1.000 1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
```

```
Layer 3 image number 10
 11  9 10  5  7  7  7  7  7
 11  9 10  5  7  7  7  7  7
 11  5 10  5  7  7  7  7  7
  9  5 10  5  7  7  7  7  7
  9  5  5  7  7  7  7  7  7
  5  5  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7

Layer 3 image number 10
1.000 0.548 0.672 0.498 0.606 1.000 1.000 1.000 1.000
0.679 0.538 0.693 0.488 0.606 1.000 1.000 1.000 1.000
0.646 0.580 0.662 0.485 0.615 1.000 1.000 1.000 1.000
0.661 0.639 0.586 0.486 0.633 1.000 1.000 1.000 1.000
0.593 0.625 0.548 0.532 0.675 1.000 1.000 1.000 1.000
0.503 0.524 0.533 0.619 0.769 1.000 1.000 1.000 1.000
0.611 0.608 0.657 0.752 1.000 1.000 1.000 1.000 1.000
0.817 0.802 0.905 1.000 1.000 1.000 1.000 1.000 1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000

Layer 3 image number 11
  8  8  5  5  7  7  7  7  7
  8  8  5  5  7  7  7  7  7
  8  8  5  5  7  7  7  7  7
  8  5  5  5  7  7  7  7  7
  9  5  5  7  7  7  7  7  7
  5  5  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7

Layer 3 image number 11
0.937 0.700 0.674 0.637 0.513 0.676 1.000 1.000 1.000
0.779 0.688 0.696 0.620 0.530 0.699 1.000 1.000 1.000
0.747 0.670 0.701 0.578 0.574 0.779 1.000 1.000 1.000
0.739 0.666 0.653 0.505 0.640 1.000 1.000 1.000 1.000
0.648 0.642 0.540 0.580 0.744 1.000 1.000 1.000 1.000
0.572 0.520 0.563 0.692 1.000 1.000 1.000 1.000 1.000
0.552 0.602 0.691 0.829 1.000 1.000 1.000 1.000 1.000
0.711 0.784 0.993 1.000 1.000 1.000 1.000 1.000 1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000

Layer 3 image number 12
  5  5  7  7  7  7  7  7  7
  5  5  7  7  7  7  7  7  7
  5  5  5  7  7  7  7  7  7
  8  5  5  7  7  7  7  7  7
  9  5  5  7  7  7  7  7  7
  5  5  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7

Layer 3 image number 12
0.673 0.616 0.544 0.757 1.000 1.000 1.000 1.000 1.000
0.654 0.621 0.514 0.685 1.000 1.000 1.000 1.000 1.000
0.657 0.661 0.520 0.643 0.891 1.000 1.000 1.000 1.000
0.668 0.681 0.569 0.594 0.783 1.000 1.000 1.000 1.000
0.635 0.645 0.550 0.597 0.786 1.000 1.000 1.000 1.000
0.592 0.555 0.533 0.658 0.903 1.000 1.000 1.000 1.000
0.505 0.544 0.620 0.751 1.000 1.000 1.000 1.000 1.000
0.645 0.681 0.764 0.993 1.000 1.000 1.000 1.000 1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
```

```
Layer 3 image number 13
  5  5  5  7  7  7  7  7  7
  5  5  5  7  7  7  7  7  7
  8  5  5  7  7  7  7  7  7
  8  5  5  7  7  7  7  7  7
  9  5  5  7  7  7  7  7  7
  5  5  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7

Layer 3 image number 13
0.711 0.670 0.541 0.620 1.000 1.000 1.000 1.000 1.000
0.650 0.657 0.534 0.622 1.000 1.000 1.000 1.000 1.000
0.658 0.670 0.547 0.614 0.891 1.000 1.000 1.000 1.000
0.677 0.683 0.580 0.582 0.783 1.000 1.000 1.000 1.000
0.647 0.642 0.549 0.594 0.786 1.000 1.000 1.000 1.000
0.588 0.550 0.537 0.658 0.903 1.000 1.000 1.000 1.000
0.509 0.548 0.623 0.751 1.000 1.000 1.000 1.000 1.000
0.645 0.681 0.764 0.993 1.000 1.000 1.000 1.000 1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000

Layer 3 image number 14
  4  5  5  5  7  7  7. 7  7
  8  5  5  5  7  7  7  7  7
  8  5  5  7  7  7  7  7  7
  8  5  5  7  7  7  7  7  7
  9  5  5  7  7  7  7  7  7
  5  5  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7

Layer 3 image number 14
0.721 0.635 0.598 0.505 0.631 1.000 1.000 1.000 1.000
0.694 0.640 0.610 0.506 0.633 1.000 1.000 1.000 1.000
0.743 0.668 0.623 0.499 0.647 1.000 1.000 1.000 1.000
0.732 0.697 0.610 0.528 0.682 1.000 1.000 1.000 1.000
0.659 0.644 0.534 0.591 0.769 1.000 1.000 1.000 1.000
0.580 0.523 0.561 0.692 1.000 1.000 1.000 1.000 1.000
0.538 0.594 0.691 0.829 1.000 1.000 1.000 1.000 1.000
0.686 0.764 0.993 1.000 1.000 1.000 1.000 1.000 1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000

Layer 3 image number 15
  3  9  5  5  7  7  7. 7  7
  8  9  5  5  7  7  7  7  7
  8  9  5  5  7  7  7  7  7
  9  9  5  5  7  7  7  7  7
  9  5  5  7  7  7  7  7  7
  5  5  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7

Layer 3 image number 15
0.823 0.622 0.596 0.568 0.566 0.891 1.000 1.000 1.000
0.703 0.649 0.648 0.605 0.545 0.783 1.000 1.000 1.000
0.722 0.690 0.676 0.610 0.552 0.786 1.000 1.000 1.000
0.690 0.677 0.687 0.569 0.588 0.903 1.000 1.000 1.000
0.631 0.645 0.596 0.503 0.636 1.000 1.000 1.000 1.000
0.514 0.531 0.518 0.598 0.735 1.000 1.000 1.000 1.000
0.611 0.608 0.657 0.752 1.000 1.000 1.000 1.000 1.000
0.817 0.802 0.905 1.000 1.000 1.000 1.000 1.000 1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
```

```
Layer 3 image number 16
  1  5  5  7  7  7  7  7  7
  8  5  5  7  7  7  7  7  7
  8  5  5  7  7  7  7  7  7
  9  5  5  7  7  7  7  7  7
  5  5  5  7  7  7  7  7  7
  5  5  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7

Layer 3 image number 16
1.000 0.659 0.588 0.568 0.742 1.000 1.000 1.000 1.000
0.575 0.631 0.610 0.536 0.696 1.000 1.000 1.000 1.000
0.699 0.683 0.631 0.520 0.676 1.000 1.000 1.000 1.000
0.644 0.664 0.607 0.538 0.699 1.000 1.000 1.000 1.000
0.621 0.600 0.531 0.595 0.779 1.000 1.000 1.000 1.000
0.580 0.522 0.563 0.692 1.000 1.000 1.000 1.000 1.000
0.535 0.594 0.691 0.829 1.000 1.000 1.000 1.000 1.000
0.681 0.764 0.993 1.000 1.000 1.000 1.000 1.000 1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000

Layer 3 image number 17
  6  5  5  7  7  7  7  7  7
  8  5  5  7  7  7  7  7  7
  8  5  5  7  7  7  7  7  7
  8  5  5  7  7  7  7  7  7
  5  5  5  7  7  7  7  7  7
  5  5  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7

Layer 3 image number 17
0.804 0.680 0.581 0.571 0.742 1.000 1.000 1.000 1.000
0.631 0.661 0.611 0.536 0.696 1.000 1.000 1.000 1.000
0.707 0.675 0.633 0.520 0.676 1.000 1.000 1.000 1.000
0.672 0.689 0.609 0.538 0.699 1.000 1.000 1.000 1.000
0.634 0.625 0.532 0.595 0.779 1.000 1.000 1.000 1.000
0.580 0.523 0.561 0.692 1.000 1.000 1.000 1.000 1.000
0.538 0.594 0.691 0.829 1.000 1.000 1.000 1.000 1.000
0.686 0.764 0.993 1.000 1.000 1.000 1.000 1.000 1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000

Layer 3 image number 18
  9  5  5  7  7  7  7  7  7
  9  5  5  7  7  7  7  7  7
  9 10  5  7  7  7  7  7  7
  5 10  5  7  7  7  7  7  7
  5  5  7  7  7  7  7  7  7
  5  7  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7

Layer 3 image number 18
0.951 0.676 0.610 0.508 0.675 1.000 1.000 1.000 1.000
0.707 0.645 0.578 0.555 0.769 1.000 1.000 1.000 1.000
0.641 0.602 0.531 0.595 1.000 1.000 1.000 1.000 1.000
0.658 0.578 0.499 0.627 1.000 1.000 1.000 1.000 1.000
0.625 0.548 0.532 0.675 1.000 1.000 1.000 1.000 1.000
0.524 0.533 0.619 0.769 1.000 1.000 1.000 1.000 1.000
0.608 0.657 0.752 1.000 1.000 1.000 1.000 1.000 1.000
0.802 0.905 1.000 1.000 1.000 1.000 1.000 1.000 1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
```

```
Layer 3 image number 19
  4  5  5  5  7  7  7  7  7
  8  5  5  5  7  7  7  7  7
  8  5  5  5  7  7  7  7  7
  8  5  5  7  7  7  7  7  7
  9  5  5  7  7  7  7  7  7
  5  5  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7

Layer 3 image number 19
1.000 0.799 0.852 0.895 1.000 1.000 1.000 1.000 1.000
0.839 0.843 0.872 1.000 1.000 1.000 1.000 1.000 1.000
0.847 0.837 0.858 0.961 1.000 1.000 1.000 1.000 1.000
0.890 0.844 0.860 0.943 1.000 1.000 1.000 1.000 1.000
0.842 0.841 0.895 1.000 1.000 1.000 1.000 1.000 1.000
0.858 0.898 0.975 1.000 1.000 1.000 1.000 1.000 1.000
0.938 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000

Layer 3 image number 20
 11  9 10  5  7  7  7  7  7
 11  9 10  5  7  7  7  7  7
  3  5 10  5  7  7  7  7  7
  9  5  5  5  7  7  7  7  7
  9  5  5  7  7  7  7  7  7
  5  5  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7

Layer 3 image number 20
0.821 0.557 0.685 0.498 0.606 1.000 1.000 1.000 1.000
0.675 0.551 0.668 0.493 0.606 1.000 1.000 1.000 1.000
0.662 0.602 0.631 0.496 0.615 1.000 1.000 1.000 1.000
0.715 0.654 0.585 0.496 0.633 1.000 1.000 1.000 1.000
0.695 0.652 0.570 0.524 0.675 1.000 1.000 1.000 1.000
0.568 0.556 0.509 0.604 0.769 1.000 1.000 1.000 1.000
0.541 0.558 0.615 0.724 1.000 1.000 1.000 1.000 1.000
0.691 0.711 0.784 0.993 1.000 1.000 1.000 1.000 1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000

Layer 3 image number 21
  7  7  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7
  5  5  7  7  7  7  7  7  7
  5  5  5  7  7  7  7  7  7
  8  5  5  5  7  7  7  7  7
  8  8  5  5  7  7  7  7  7
  8  8  5  5  7  7  7  7  7
  8  8  5  5  7  7  7  7  7

Layer 3 image number 21
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
0.698 0.767 0.909 1.000 1.000 1.000 1.000 1.000 1.000
0.542 0.592 0.683 0.820 1.000 1.000 1.000 1.000 1.000
0.576 0.529 0.556 0.687 1.000 1.000 1.000 1.000 1.000
0.603 0.641 0.552 0.575 0.740 1.000 1.000 1.000 1.000
0.722 0.643 0.639 0.515 0.636 1.000 1.000 1.000 1.000
0.723 0.648 0.685 0.581 0.571 0.776 1.000 1.000 1.000
0.749 0.662 0.676 0.618 0.527 0.696 1.000 1.000 1.000
1.000 0.696 0.681 0.634 0.513 0.676 1.000 1.000 1.000
```

```
Layer 3 image number 22
  7  7  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7
  5  7  7  7  7  7  7  7  7
  5  5  7  7  7  7  7  7  7
 10  5  7  7  7  7  7  7  7
 10  5  7  7  7  7  7  7  7
 10  5  7  7  7  7  7  7  7

Layer 3 image number 22
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
0.869 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
0.633 0.733 1.000 1.000 1.000 1.000 1.000 1.000 1.000
0.513 0.607 0.767 1.000 1.000 1.000 1.000 1.000 1.000
0.573 0.528 0.675 1.000 1.000 1.000 1.000 1.000 1.000
0.601 0.505 0.634 1.000 1.000 1.000 1.000 1.000 1.000
0.635 0.505 0.615 1.000 1.000 1.000 1.000 1.000 1.000
0.697 0.492 0.606 1.000 1.000 1.000 1.000 1.000 1.000
1.000 0.494 0.582 0.909 1.000 1.000 1.000 1.000 1.000

Layer 3 image number 23
  7  7  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7
  5  5  7  7  7  7  7  7  7
  5  5  5  7  7  7  7  7  7
  8  5  5  7  7  7  7  7  7
  8  5  5  7  7  7  7  7  7
  9  5  5  7  7  7  7  7  7
  5  5  7  7  7  7  7  7  7

Layer 3 image number 23
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
0.673 0.749 0.909 1.000 1.000 1.000 1.000 1.000 1.000
0.528 0.585 0.683 0.820 1.000 1.000 1.000 1.000 1.000
0.583 0.529 0.562 0.698 1.000 1.000 1.000 1.000 1.000
0.618 0.638 0.529 0.623 0.891 1.000 1.000 1.000 1.000
0.697 0.698 0.589 0.578 0.783 1.000 1.000 1.000 1.000
0.692 0.690 0.590 0.580 0.786 1.000 1.000 1.000 1.000
0.651 0.675 0.526 0.627 0.903 1.000 1.000 1.000 1.000
1.000 0.630 0.517 0.661 1.000 1.000 1.000 1.000 1.000

Layer 3 image number 24
  7  7  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7
  5  5  7  7  7  7  7  7  7
  1  5  5  7  7  7  7  7  7
 11  5  5  7  7  7  7  7  7
  8  5  5  7  7  7  7  7  7
  8  5  5  5  7  7  7  7  7
  2  5  5  5  7  7  7  7  7

Layer 3 image number 24
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
0.673 0.749 0.909 1.000 1.000 1.000 1.000 1.000 1.000
0.528 0.585 0.683 0.820 1.000 1.000 1.000 1.000 1.000
0.582 0.531 0.556 0.687 1.000 1.000 1.000 1.000 1.000
0.660 0.616 0.541 0.591 0.776 1.000 1.000 1.000 1.000
0.582 0.634 0.611 0.530 0.689 1.000 1.000 1.000 1.000
0.711 0.672 0.636 0.501 0.653 1.000 1.000 1.000 1.000
0.625 0.622 0.632 0.507 0.638 1.000 1.000 1.000 1.000
1.000 0.588 0.614 0.510 0.631 1.000 1.000 1.000 1.000
```

```
Layer 3 image number 25
  7  7  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7
  5  5  7  7  7  7  7  7  7
  5  5  5  7  7  7  7  7  7
  8  5  5  5  7  7  7  7  7
 11  5 10  5  7  7  7  7  7
 11  9 10  5  7  7  7  7  7
  3  9  5  5  7  7  7  7  7

Layer 3 image number 25
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
0.787 0.778 0.869 1.000 1.000 1.000 1.000 1.000 1.000
0.577 0.583 0.633 0.733 1.000 1.000 1.000 1.000 1.000
0.544 0.546 0.512 0.607 0.767 1.000 1.000 1.000 1.000
0.594 0.642 0.575 0.528 0.675 1.000 1.000 1.000 1.000
0.660 0.645 0.603 0.505 0.634 1.000 1.000 1.000 1.000
0.683 0.614 0.669 0.518 0.607 1.000 1.000 1.000 1.000
0.674 0.600 0.630 0.535 0.584 1.000 1.000 1.000 1.000
1.000 0.620 0.596 0.568 0.566 0.891 1.000 1.000 1.000

Layer 3 image number 26
  7  7  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7
  5  5  7  7  7  7  7  7  7
  8  5  5  7  7  7  7  7  7
  8  5  5  7  7  7  7  7  7
  8  5  5  7  7  7  7  7  7
  8  5  5  7  7  7  7  7  7
  1  5  5  7  7  7  7  7  7

Layer 3 image number 26
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
0.631 0.667 0.749 0.909 1.000 1.000 1.000 1.000 1.000
0.498 0.538 0.613 0.743 1.000 1.000 1.000 1.000 1.000
0.595 0.558 0.530 0.654 0.891 1.000 1.000 1.000 1.000
0.606 0.646 0.553 0.594 0.783 1.000 1.000 1.000 1.000
0.678 0.691 0.574 0.589 0.786 1.000 1.000 1.000 1.000
0.657 0.708 0.556 0.608 0.840 1.000 1.000 1.000 1.000
0.669 0.676 0.563 0.593 0.792 1.000 1.000 1.000 1.000
1.000 0.658 0.584 0.568 0.742 1.000 1.000 1.000 1.000

Layer 3 image number 27
  7  7  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7
  5  5  7  7  7  7  7  7  7
  8  5  5  7  7  7  7  7  7
  8  5  5  7  7  7  7  7  7
  8  5  5  7  7  7  7  7  7
  8  5  5  7  7  7  7  7  7
  1  5  5  7  7  7  7  7  7

Layer 3 image number 27
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
0.635 0.667 0.749 0.909 1.000 1.000 1.000 1.000 1.000
0.501 0.538 0.613 0.743 1.000 1.000 1.000 1.000 1.000
0.592 0.558 0.530 0.654 0.891 1.000 1.000 1.000 1.000
0.604 0.646 0.553 0.594 0.783 1.000 1.000 1.000 1.000
0.680 0.691 0.574 0.589 0.786 1.000 1.000 1.000 1.000
0.662 0.708 0.556 0.608 0.840 1.000 1.000 1.000 1.000
0.670 0.676 0.563 0.593 0.792 1.000 1.000 1.000 1.000
1.000 0.658 0.584 0.568 0.742 1.000 1.000 1.000 1.000
```

```
Layer 3 image number 28
  7  7  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7
  5  7  7  7  7  7  7  7  7
  5  5  5  7  7  7  7  7  7
  8  5  5  7  7  7  7  7  7
  8  5  5  5  7  7  7  7  7
  8  5 10  5  7  7  7  7  7
  9  5  5  5  7  7  7  7  7
  9  5  5  7  7  7  7  7  7

Layer 3 image number 28
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
0.631 0.667 0.749 0.909 1.000 1.000 1.000 1.000 1.000
0.493 0.527 0.596 0.717 1.000 1.000 1.000 1.000 1.000
0.593 0.572 0.505 0.601 0.767 1.000 1.000 1.000 1.000
0.619 0.654 0.588 0.525 0.675 1.000 1.000 1.000 1.000
0.649 0.643 0.602 0.504 0.637 1.000 1.000 1.000 1.000
0.671 0.639 0.597 0.504 0.629 1.000 1.000 1.000 1.000
0.697 0.653 0.595 0.500 0.637 1.000 1.000 1.000 1.000
1.000 0.666 0.613 0.508 0.675 1.000 1.000 1.000 1.000

Layer 3 image number 29
  7  7  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7
  5  5  7  7  7  7  7  7  7
  1  5  5  7  7  7  7  7  7
 11  5  5  7  7  7  7  7  7
  8  5  5  7  7  7  7  7  7
  8  5  5  5  7  7  7  7  7
  2  5  5  5  7  7  7  7  7

Layer 3 image number 29
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
0.673 0.749 0.909 1.000 1.000 1.000 1.000 1.000 1.000
0.528 0.585 0.683 0.820 1.000 1.000 1.000 1.000 1.000
0.582 0.531 0.556 0.687 1.000 1.000 1.000 1.000 1.000
0.661 0.616 0.541 0.591 0.776 1.000 1.000 1.000 1.000
0.582 0.634 0.611 0.530 0.689 1.000 1.000 1.000 1.000
0.710 0.672 0.636 0.501 0.653 1.000 1.000 1.000 1.000
0.624 0.622 0.632 0.507 0.638 1.000 1.000 1.000 1.000
1.000 0.588 0.614 0.510 0.631 1.000 1.000 1.000 1.000

Layer 3 image number 30
  7  7  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7
  5  5  7  7  7  7  7  7  7
  6  5  5  7  7  7  7  7  7
  8  5  5  7  7  7  7  7  7
  8  5  5  5  7  7  7  7  7
 11  9 10  5  7  7  7  7  7
 11  9 10  5  7  7  7  7  7

Layer 3 image number 30
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
0.673 0.749 0.909 1.000 1.000 1.000 1.000 1.000 1.000
0.528 0.585 0.683 0.820 1.000 1.000 1.000 1.000 1.000
0.582 0.531 0.556 0.687 1.000 1.000 1.000 1.000 1.000
0.600 0.640 0.545 0.586 0.767 1.000 1.000 1.000 1.000
0.655 0.669 0.612 0.518 0.675 1.000 1.000 1.000 1.000
0.659 0.633 0.606 0.507 0.634 1.000 1.000 1.000 1.000
0.672 0.597 0.635 0.506 0.615 1.000 1.000 1.000 1.000
1.000 0.548 0.672 0.498 0.606 1.000 1.000 1.000 1.000
```

```
Layer 3 image number 31
  7  7  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7
  7  7  7  7  7  7  7  7  7

Layer 3 image number 31
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000

Layer 3 image number 32
  8  8  5  5  5  9 11  9 10
  8  8  5  5  5  9 11  5 10
  8  8  5  5  9  9  3  9  5
  8  8  8  8  8  8  3  8  8
  8  8  8  8  8  8  8  8  6
  8  8  5  5  8  8  8  8 11
  8  5  5  5  9  8  8  8  8
  9  5  5  5  5  9  8  8  8
  5  5  5  5  5  9  8  8  2

Layer 3 image number 32
1.000 0.698 0.674 0.664 0.622 0.601 0.666 0.530 1.000
0.796 0.698 0.676 0.654 0.609 0.595 0.646 0.552 0.619
0.747 0.709 0.652 0.643 0.607 0.593 0.672 0.604 0.582
0.783 0.713 0.668 0.633 0.626 0.614 0.672 0.665 0.636
0.786 0.681 0.666 0.639 0.646 0.637 0.683 0.638 0.734
0.729 0.673 0.653 0.636 0.645 0.655 0.683 0.630 0.608
0.693 0.661 0.663 0.626 0.644 0.669 0.697 0.753 0.701
0.655 0.693 0.620 0.597 0.608 0.647 0.700 0.729 0.623
1.000 0.652 0.589 0.575 0.606 0.648 0.705 0.702 1.000
```

Results A3.6 Layer 3 results

The features classified by the neurons are shown in Table A3.6.

Feature Neuron	Learnt Feature
1	Number 5, 6
2	Number 3, 8
3	Number 4
4	Number 3, 8*
5	Number 2
6	Number 6*
7	Background
8	Number 0
9	Number 7
10	Number 1
11	Number 9

Table A3.6 Features classified by the neurons

The values denoted by * were determined from the full output results.

Comparing the classification performance for the single and triple layers shows that the triple layer had the unfortunate characteristic of two different feature neurons classifying the same feature. The Numbers 5 and 6, and 3 and 8 are considered to be the same. This is caused by the combined tolerance of the three layers reducing the discrimination of the network.

When the triple layer network was classifying the noisy images, there was a degradation performance when compared with the single layer. On examination of the results, it was discovered that the low-level layers were very sensitive to the noise and misclassified the

features. This effect was caused by the smaller number of pixels in the receptive field; hence, the noise in the image had a greater effect.

The triple layer network classified the translated images with very similar output levels as when classifying the training set.

APPENDIX 4 – ARTIFICIAL RETINA AND GREY SCALE RESULTS

This appendix demonstrates the behaviour of the artificial retina and gives the results for the grey scale classification. When testing the retina the following values were used:

- Matrix conductance 1S
- Receptor conductance 1S
- Residual 0.1

TESTING THE RETINA

The retina was tested with a 16 row × 16 column retina having a step function as shown in Results A4.1.

```
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
```

Results A4.1 Input step function

This shows a step from values 1.0 to 0.0, and when passed through the retina the output is given by Results A4.2.

```
0.5089 0.5089 0.5089 0.4823 0.5092 0.5407 0.6085 0.7805 0.2284 0.4004
0.4682 0.4997 0.5266 0.5000 0.5000 0.5000
 0.5089 0.5089 0.5089 0.4823 0.5092 0.5407 0.6085 0.7805 0.2284 0.4004
0.4682 0.4997 0.5266 0.5000 0.5000 0.5000
 0.5089 0.5089 0.5089 0.4823 0.5092 0.5407 0.6085 0.7805 0.2284 0.4004
0.4682 0.4997 0.5266 0.5000 0.5000 0.5000
 0.5089 0.5089 0.5089 0.4823 0.5092 0.5407 0.6085 0.7805 0.2284 0.4004
0.4682 0.4997 0.5266 0.5000 0.5000 0.5000
 0.5089 0.5089 0.5089 0.4823 0.5092 0.5407 0.6085 0.7805 0.2284 0.4004
0.4682 0.4997 0.5266 0.5000 0.5000 0.5000
 0.5089 0.5089 0.5089 0.4823 0.5092 0.5407 0.6085 0.7805 0.2284 0.4004
0.4682 0.4997 0.5266 0.5000 0.5000 0.5000
 0.5089 0.5089 0.5089 0.4823 0.5092 0.5407 0.6085 0.7805 0.2284 0.4004
0.4682 0.4997 0.5266 0.5000 0.5000 0.5000
 0.5089 0.5089 0.5089 0.4823 0.5092 0.5407 0.6085 0.7805 0.2284 0.4004
0.4682 0.4997 0.5266 0.5000 0.5000 0.5000
 0.5089 0.5089 0.5089 0.4823 0.5092 0.5407 0.6085 0.7805 0.2284 0.4004
0.4682 0.4997 0.5266 0.5000 0.5000 0.5000
 0.5089 0.5089 0.5089 0.4823 0.5092 0.5407 0.6085 0.7805 0.2284 0.4004
0.4682 0.4997 0.5266 0.5000 0.5000 0.5000
 0.5089 0.5089 0.5089 0.4823 0.5092 0.5407 0.6085 0.7805 0.2284 0.4004
0.4682 0.4997 0.5266 0.5000 0.5000 0.5000
 0.5089 0.5089 0.5089 0.4823 0.5092 0.5407 0.6085 0.7805 0.2284 0.4004
0.4682 0.4997 0.5266 0.5000 0.5000 0.5000
```

Results A4.2 Retina output

The results displayed graphically in Figure A4.1.

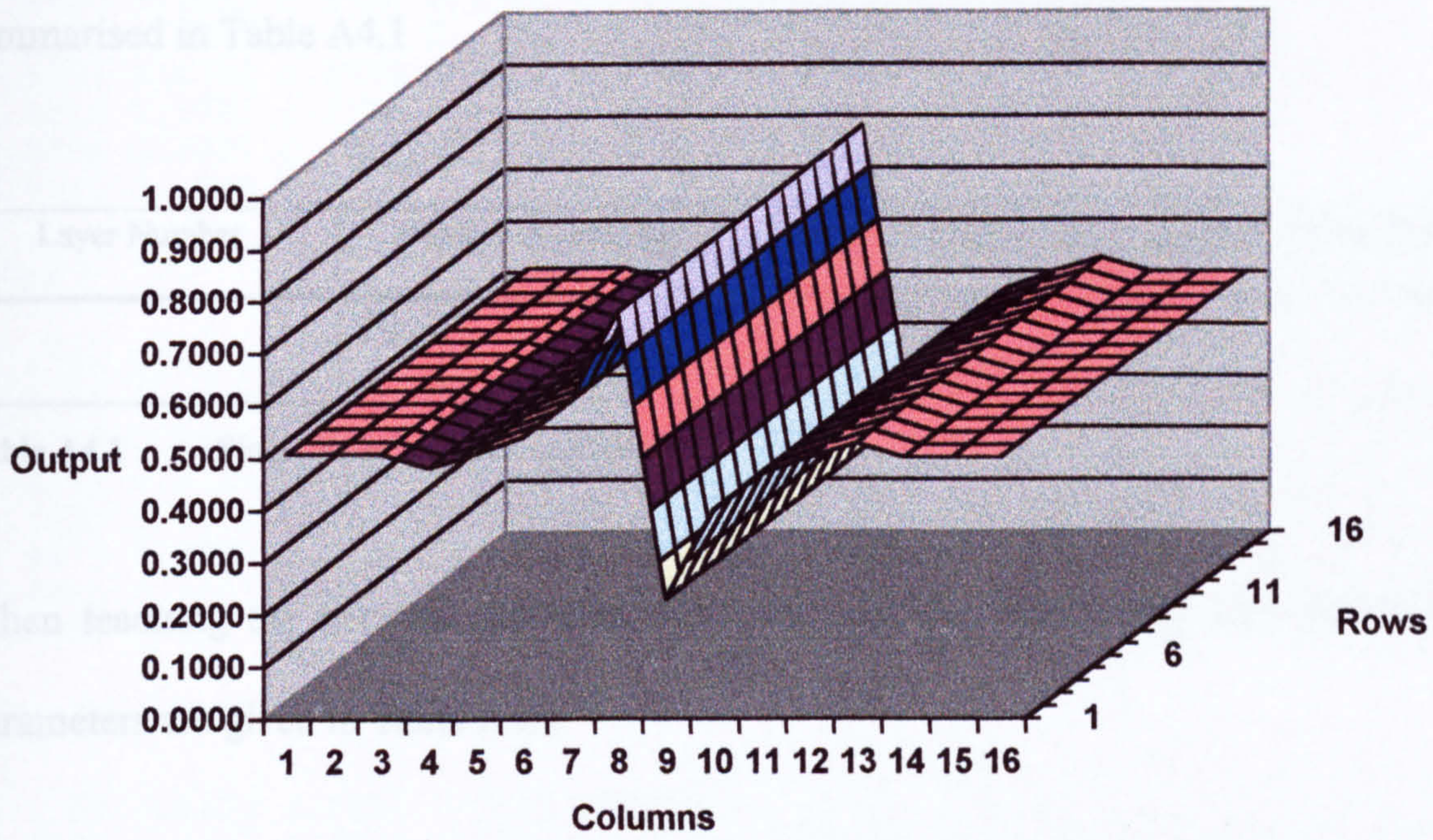


Figure A4.1 Retina output for step input

The response to a step function behaves very favourably with the results produced by Mead [Mead and Mahowald 1988].

IMAGE CLASSIFICATION

The classification of two grey scale images was tested by considering images with a size of 6 rows × 8 columns. The image set consisted of 20 images with the first 10 having brightness levels of 0.1 and 0.2, and the second 10 having brightness values of 0.5 and 1.0. Both groups of images showed the same images, the numbers from 0 to 9, with different brightness levels. The contrast ratio between the two images was the same at 2.0 but the second group of images had five times the brightness.

A single layer network with 10 features was used to classify the images. The receptive field of the neurons was the same size as the images so one neuron was required to classify the image and there was no possibility of shift invariance. The network parameters are summarised in Table A4.1

Layer Number	Number of Features	Receptive Field Rows	Receptive Field Columns
1	10	6	8

Table A4.1 Single layer network configuration

When teaching the network, the first 10 images are the training set and the teaching parameters are given in Table A4.2.

Layer	Tiredness reduction	Tiredness enhancement	Atrophy	Tolerance	Learning rate	Teaching presentations
1	0.02	0.75	0.25	0.02	0.5	25

Table A4.2 Single layer network teaching and neuron run parameters

The images used are shown in Figure A4.2.

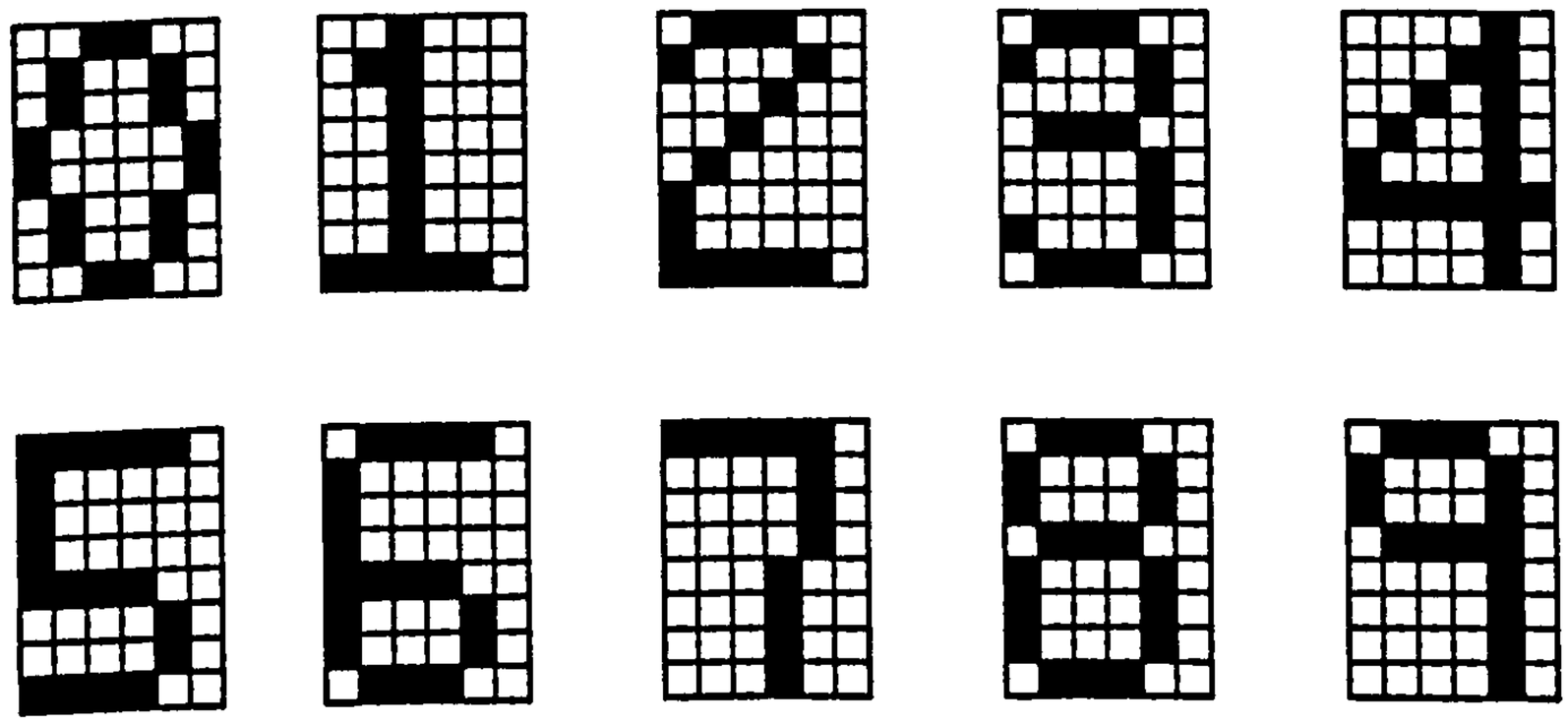


Figure A4.2 Images used

The outputs from the retina are shown in Results A4.1.

```
0.4753 0.4706 0.5585 0.5585 0.4706 0.4753
0.4798 0.6037 0.4210 0.4210 0.6037 0.4798
0.4210 0.6037 0.4798 0.4798 0.6037 0.4210
0.5585 0.4706 0.4753 0.4753 0.4706 0.5585
0.5585 0.4706 0.4753 0.4753 0.4706 0.5585
0.4210 0.6037 0.4798 0.4798 0.6037 0.4210
0.4798 0.6037 0.4210 0.4210 0.6037 0.4798
0.4753 0.4706 0.5585 0.5585 0.4706 0.4753
```

Image 1

```
0.4726 0.4518 0.5728 0.4444 0.5010 0.5142
0.4948 0.5744 0.5720 0.4699 0.4831 0.5165
0.4845 0.4405 0.5986 0.4305 0.5010 0.5142
0.4921 0.4745 0.5643 0.4725 0.4940 0.5142
0.5232 0.4530 0.6179 0.4593 0.4947 0.5235
0.4875 0.4772 0.5849 0.4587 0.4802 0.5255
0.4485 0.4201 0.5412 0.4203 0.4769 0.4825
0.5558 0.5177 0.5263 0.5316 0.5799 0.4716
```

Image 2

```
0.4198 0.5673 0.5482 0.5466 0.4541 0.4879
0.6205 0.4363 0.4206 0.4328 0.5903 0.5025
0.4915 0.4738 0.4451 0.6042 0.4804 0.4890
0.4708 0.4860 0.6263 0.4625 0.4968 0.4995
0.4574 0.6040 0.4852 0.5076 0.4849 0.5309
0.5703 0.4542 0.4935 0.4879 0.5288 0.5181
0.5281 0.4219 0.4281 0.4664 0.4763 0.4921
0.4912 0.5133 0.5508 0.5449 0.5724 0.4615
```

Image 3

```
0.4259 0.5537 0.5578 0.5463 0.4529 0.4927
0.6062 0.4549 0.4180 0.4409 0.5890 0.4924
0.4726 0.4552 0.4504 0.4367 0.5944 0.4796
0.4770 0.6198 0.5983 0.5717 0.4476 0.4703
0.4897 0.4820 0.4861 0.4255 0.5905 0.4724
0.4966 0.4971 0.4651 0.4746 0.5917 0.4644
0.6022 0.4569 0.4496 0.4301 0.5958 0.4796
0.4203 0.5505 0.5490 0.5487 0.4449 0.4983
```

Image 4

```
0.5254 0.5086 0.4805 0.4350 0.5743 0.4595
0.5057 0.4999 0.4727 0.5597 0.5728 0.4557
0.4814 0.4903 0.6032 0.4203 0.5896 0.4527
0.4544 0.5926 0.4546 0.4496 0.5718 0.4683
0.5378 0.4217 0.4454 0.3997 0.5705 0.4262
0.5499 0.5723 0.5685 0.5672 0.5553 0.5468
0.4516 0.4553 0.4784 0.4209 0.5482 0.4277
0.5175 0.5279 0.5026 0.4653 0.6023 0.4552
```

Image 5

```
0.4961 0.5235 0.5499 0.5564 0.5780 0.4592
0.5292 0.4189 0.4520 0.4508 0.4830 0.4866
0.5538 0.4436 0.4724 0.5154 0.4969 0.5342
0.5338 0.4120 0.4893 0.4703 0.4973 0.5092
0.5622 0.5965 0.5948 0.6077 0.4752 0.4762
0.4520 0.4527 0.4558 0.4384 0.6045 0.4715
0.4482 0.4443 0.4186 0.4454 0.5853 0.4907
0.5499 0.5538 0.5622 0.5496 0.4634 0.4762
```

Image 6

```
0.4211 0.5610 0.5523 0.5464 0.5711 0.4613
0.5683 0.4450 0.4543 0.4469 0.4832 0.4895
0.5602 0.4543 0.4673 0.5172 0.4988 0.5330
0.5178 0.4105 0.5011 0.4711 0.4995 0.5098
0.5116 0.5625 0.5910 0.6113 0.4717 0.4799
0.5382 0.4158 0.4417 0.4427 0.6028 0.4723
0.5687 0.4397 0.4162 0.4427 0.5852 0.4905
0.4266 0.5683 0.5765 0.5512 0.4604 0.4809
```

Image 7

```
0.5411 0.5473 0.5428 0.5273 0.5692 0.4408
0.4489 0.4487 0.4453 0.4138 0.5864 0.4484
0.5054 0.5259 0.4593 0.4693 0.5873 0.4702
0.5241 0.5089 0.4951 0.4669 0.5919 0.4722
0.5076 0.5103 0.4676 0.5838 0.4712 0.4780
0.5138 0.4808 0.4785 0.5757 0.4690 0.4925
0.5138 0.4932 0.4469 0.6001 0.4447 0.5116
0.5138 0.4932 0.4552 0.5854 0.4593 0.5034
```

Image 8

```
0.4222 0.5605 0.5661 0.5458 0.4517 0.4929
0.5791 0.4411 0.4157 0.4456 0.5899 0.4916
0.5640 0.4398 0.4547 0.4361 0.5949 0.4797
0.4016 0.5831 0.6066 0.5803 0.4482 0.4693
0.5440 0.4434 0.4876 0.4314 0.5895 0.4733
0.5620 0.4601 0.4636 0.4766 0.5954 0.4626
0.5640 0.4398 0.4547 0.4361 0.5949 0.4797
0.4201 0.5626 0.5573 0.5479 0.4441 0.4984
```

Image 9

```
0.4256 0.5573 0.5668 0.5551 0.4581 0.4918
0.5683 0.4400 0.4251 0.4354 0.5855 0.4922
0.5679 0.4387 0.4413 0.4293 0.5772 0.4614
0.4475 0.5958 0.5769 0.5653 0.5653 0.4487
0.4826 0.4750 0.4836 0.4139 0.5746 0.4529
0.5120 0.5202 0.4764 0.4611 0.5776 0.4649
0.5116 0.5118 0.5082 0.4597 0.5896 0.4543
0.5028 0.5117 0.5010 0.4596 0.5826 0.4543
```

Image 10

```
0.4571 0.4297 0.6135 0.6135 0.4297 0.4571
0.4303 0.6477 0.4223 0.4223 0.6477 0.4303
0.4223 0.6477 0.4303 0.4303 0.6477 0.4223
0.6135 0.4297 0.4571 0.4571 0.4297 0.6135
0.6135 0.4297 0.4571 0.4571 0.4297 0.6135
0.4223 0.6477 0.4303 0.4303 0.6477 0.4223
0.4303 0.6477 0.4223 0.4223 0.6477 0.4303
0.4571 0.4297 0.6135 0.6135 0.4297 0.4571
```

Image 11

```
0.4469 0.4269 0.6276 0.4352 0.4712 0.5158
0.4479 0.6504 0.6170 0.4409 0.4799 0.5099
0.4662 0.4346 0.6312 0.4509 0.4635 0.5158
0.4693 0.4497 0.6479 0.4372 0.4837 0.5158
0.4775 0.4793 0.6348 0.4658 0.4718 0.5231
0.4542 0.4380 0.6207 0.4320 0.4699 0.4985
0.4116 0.3828 0.5827 0.3999 0.4359 0.4709
0.5812 0.5623 0.5536 0.5804 0.6174 0.4527
```

Image 12

```
0.4123 0.5935 0.5818 0.6102 0.4258 0.4616
0.6318 0.4155 0.4183 0.3957 0.6617 0.4637
0.4542 0.4566 0.4156 0.6508 0.4520 0.4924
0.4607 0.4416 0.6763 0.4764 0.4842 0.4915
0.4344 0.6667 0.4731 0.4784 0.5107 0.5101
0.6108 0.4195 0.4360 0.4733 0.4776 0.5221
0.5546 0.3673 0.4055 0.4184 0.4711 0.4672
0.5209 0.5413 0.5707 0.6219 0.6306 0.4405
```

Image 13

```
0.3848 0.5819 0.5882 0.5704 0.4266 0.4881
0.6630 0.4297 0.3726 0.4080 0.6364 0.4877
0.4571 0.4302 0.4227 0.4015 0.6448 0.4679
0.4639 0.6841 0.6508 0.6096 0.4185 0.4535
0.4835 0.4716 0.4778 0.3842 0.6387 0.4567
0.4942 0.4949 0.4454 0.4602 0.6406 0.4444
0.6569 0.4328 0.4215 0.3913 0.6470 0.4679
0.3762 0.5769 0.5746 0.5742 0.4143 0.4968
```

Image 14

```
0.5240 0.4908 0.4575 0.4159 0.6241 0.4193
0.4902 0.5056 0.4338 0.6323 0.6153 0.4241
0.4765 0.4434 0.6547 0.4190 0.6207 0.4250
0.4159 0.6422 0.4032 0.4030 0.6156 0.4093
0.5989 0.3991 0.4011 0.4009 0.5895 0.4004
0.5990 0.6098 0.6110 0.5876 0.5930 0.5863
0.4413 0.4582 0.4253 0.4225 0.6168 0.3848
0.4944 0.4878 0.4752 0.4455 0.6191 0.4194
```

Image 15

```
0.4925 0.5348 0.5758 0.5859 0.6192 0.4362
0.5437 0.3737 0.4249 0.4231 0.4730 0.4784
0.5817 0.4119 0.4564 0.5231 0.4944 0.5521
0.5508 0.3630 0.4827 0.4533 0.4952 0.5134
0.5948 0.6478 0.6453 0.6651 0.4610 0.4624
0.4249 0.4261 0.4309 0.4040 0.6603 0.4552
0.4190 0.4130 0.3732 0.4147 0.6305 0.4849
0.5758 0.5817 0.5948 0.5752 0.4427 0.4624
```

Image 16

```
0.3849 0.5919 0.5975 0.6042 0.6272 0.4407
0.5993 0.4030 0.4173 0.4469 0.4493 0.4814
0.5847 0.4116 0.4449 0.4515 0.5011 0.5102
0.5594 0.3971 0.4431 0.4786 0.4971 0.4944
0.5542 0.5829 0.6186 0.6664 0.4573 0.4788
0.5671 0.3854 0.3998 0.4275 0.6482 0.4617
0.5842 0.3784 0.4123 0.3934 0.6440 0.4595
0.3933 0.6056 0.5901 0.6187 0.4308 0.4568
```

Image 17

0.5620 0.5716 0.5646 0.5406 0.6055 0.4074
0.4200 0.4197 0.4144 0.3657 0.6321 0.4191
0.5075 0.5393 0.4360 0.4517 0.6335 0.4530
0.5365 0.5129 0.4916 0.4480 0.6406 0.4561
0.5109 0.5151 0.4489 0.6281 0.4547 0.4651
0.5205 0.4694 0.4659 0.6155 0.4513 0.4874
0.5205 0.4886 0.4170 0.6533 0.4135 0.5171
0.5205 0.4886 0.4298 0.6306 0.4362 0.5043

Image 18

0.3795 0.5926 0.6013 0.5699 0.4250 0.4886
0.6214 0.4086 0.3693 0.4156 0.6380 0.4867
0.5980 0.4065 0.4295 0.4008 0.6458 0.4682
0.3475 0.6275 0.6638 0.6232 0.4196 0.4522
0.5671 0.4121 0.4804 0.3937 0.6375 0.4583
0.5950 0.4380 0.4433 0.4634 0.6465 0.4417
0.5980 0.4065 0.4295 0.4008 0.6458 0.4682
0.3761 0.5959 0.5877 0.5732 0.4132 0.4971

Image 19

0.3844 0.5874 0.6023 0.5841 0.4348 0.4868
0.6045 0.4068 0.3836 0.3996 0.6311 0.4874
0.6039 0.4047 0.4087 0.3901 0.6183 0.4397
0.4183 0.6469 0.6178 0.5999 0.5999 0.4201
0.4726 0.4609 0.4741 0.3663 0.6143 0.4266
0.5180 0.5306 0.4630 0.4393 0.6189 0.4452
0.5174 0.5177 0.5121 0.4372 0.6375 0.4289
0.5038 0.5175 0.5010 0.4370 0.6266 0.4289

Image 20

Results A4.1 Retina outputs

After teaching the network its classification performance was tested and the results are summarised in Table A4.3.

The classification performance for the training set is good, but the brighter images failed to be completely classified. By having a constant term in the logarithmic photoreceptor relationship removed, the pure logarithmic relationship between the two brightness levels was established. These results were repeated using the modified photoreceptor relationship and the results are summarised in Table A4.4.

Image Num.	Neur. 1	Neur. 2	Neur. 3	Neur. 4	Neur. 5	Neur. 6	Neur. 7	Neur. 8	Neur. 9	Neur. 10
1	0.64	0.62	0.63	0.64	0.63	0.63	1.00	0.61	0.65	0.63
2	0.65	0.64	0.67	0.63	0.64	0.65	0.63	0.64	0.63	1.00
3	0.67	0.70	0.61	0.66	0.68	0.68	0.62	1.00	0.67	0.64
4	0.76	0.65	0.65	0.63	0.83	0.63	0.64	0.67	1.00	0.63
5	0.69	0.59	1.00	0.63	0.65	0.60	0.63	0.61	0.65	0.67
6	0.63	0.80	0.60	0.66	0.63	1.00	0.63	0.68	0.63	0.64
7	0.63	1.00	0.59	0.66	0.66	0.80	0.62	0.69	0.55	0.63
8	0.65	0.66	0.63	1.00	0.64	0.67	0.64	0.66	0.64	0.63
9	0.75	0.65	0.65	0.63	1.00	0.63	0.63	0.67	0.83	0.53
10	1.00	0.63	0.69	0.64	0.76	0.63	0.64	0.67	0.76	0.64
11	0.58	0.56	0.57	0.57	0.57	0.57	0.77	0.55	0.58	0.56
12	0.59	0.59	0.60	0.58	0.58	0.60	0.58	0.60	0.58	0.79
13	0.61	0.64	0.56	0.59	0.62	0.62	0.56	0.78	0.62	0.58
14	0.68	0.58	0.59	0.56	0.74	0.56	0.58	0.60	0.79	0.56
15	0.62	0.54	0.78	0.57	0.60	0.55	0.58	0.55	0.60	0.59
16	0.57	0.73	0.54	0.60	0.58	0.80	0.57	0.62	0.57	0.58
17	0.58	0.78	0.54	0.59	0.61	0.72	0.57	0.62	0.60	0.57
18	0.60	0.62	0.59	0.83	0.59	0.62	0.60	0.61	0.59	0.58
19	0.68	0.59	0.59	0.56	0.79	0.57	0.56	0.60	0.74	0.56
20	0.80	0.58	0.63	0.58	0.71	0.57	0.59	0.61	0.70	0.58

Table A4.3 Classification results using original photoreceptor

Image Num.	Neur. 1	Neur. 2	Neur. 3	Neur. 4	Neur. 5	Neur. 6	Neur. 7	Neur. 8	Neur. 9	Neur. 10
1	0.51	0.52	0.51	0.52	0.51	0.52	1.00	0.52	0.51	0.49
2	0.52	0.53	1.00	0.51	0.52	0.51	0.52	0.52	0.53	0.55
3	0.57	0.56	0.55	0.54	0.57	0.58	0.50	0.57	0.49	1.00
4	0.75	0.53	0.50	0.52	0.56	1.00	0.52	0.66	0.52	0.58
5	0.53	0.50	0.52	0.52	0.49	0.52	0.51	0.56	1.00	0.49
6	0.53	1.00	0.53	0.56	0.72	0.54	0.53	0.52	0.50	0.55
7	0.56	0.72	0.51	0.55	1.00	0.56	0.52	0.53	0.49	0.57
8	0.52	0.57	1.52	1.00	0.56	0.54	0.54	0.54	0.53	0.55
9	1.00	0.52	0.50	0.51	0.55	0.75	0.51	0.67	0.53	0.56
10	0.67	0.52	0.52	0.53	0.53	0.67	0.52	1.00	0.57	0.57
11	0.51	0.52	0.51	0.52	0.51	0.52	1.00	0.52	0.51	0.49
12	0.51	0.53	1.00	0.51	0.52	0.51	0.52	0.52	0.53	0.55
13	0.57	0.56	0.55	0.54	0.57	0.58	0.50	0.57	0.49	1.00
14	0.75	0.53	0.50	0.52	0.56	1.00	0.52	0.66	0.52	0.58
15	0.53	0.50	0.52	0.52	0.49	0.52	0.51	0.56	1.00	0.49
16	0.53	1.00	0.53	0.56	0.72	0.54	0.52	0.52	0.50	0.55
17	0.56	0.71	0.51	0.55	1.00	0.56	0.51	0.53	0.49	0.57
18	0.53	0.58	0.52	0.85	0.57	0.54	0.54	0.54	0.54	0.55
19	0.86	0.53	0.50	0.52	0.56	0.76	0.51	0.66	0.52	0.57
20	0.67	0.52	0.52	0.53	0.53	0.67	0.52	1.00	0.57	0.56

Table A4.4 Classification results when using the modified photoreceptor

The results using the modified photoreceptor are much better than for the standard photoreceptor. Apart from the images 18 and 19 not being fully classified, all the images were classified.

APPENDIX 5 – SIZE NETWORK ALGORITHMS

PRIMITIVE LAYER

The output from a primitive layer neuron is calculated using the following algorithm:

```
Get synapse inputs for the receptive field

and sum = 0

synapse sum = 0

abs error sum = 0

total synapses = 0

FOR all synapse rows DO

BEGIN

    FOR all synapse columns DO

    BEGIN

        abs error sum = abs error sum + |input - weight|

        and sum = and sum + min(input, weight)

        synapse sum = synapse sum + weight

        total synapses = total synapses + 1

    END

END

Use these values to calculate the neuron output

    net = 1 - abs error sum / total synapses

    and net = and sum / synapse sum

    and neuron output = activation function(and net)

    neuron output = activation function(net)

    tired output = tire output(and neuron output)
```

Pseudocode A5.1 **Primitive neuron output algorithm**

SIZE LAYER

The size layer dendrites are searched using the following algorithm:

Assign neuron search direction from neuron number

SWITCH neuron search direction

CASE Horizontal

BEGIN

synapses initialised = FALSE

largest found = FALSE

column start = neuron column - 1

previous max synapse value = -1

group counter = 1

IF layer number = 1 **THEN**

BEGIN

row = neuron row

column counter = column start

WHILE (column counter >= 1

AND

NOT largest found) **DO**

BEGIN

max synapse value = -1

FOR dendrite counter = 1 TO number of dendrites

BEGIN

determine synapse array index for dendrite counter

IF NOT synapses initialise **THEN**

BEGIN

synapse array [index] = -1

END

read input value from last primitive layer at position row, column

temp synapse values [dendrite counter - 1] = input value

max synapse value = Max(max synapse value, input value)

END

synapses initialised = TRUE

IF (max synapse value > previous max synapse value) **THEN**

BEGIN

IF (max synapse value = 1) **THEN**

BEGIN

largest found = TRUE

END

END

copy values from temp synapse values to synapse array

position 1 = column start - column counter

previous max synapse value = max synapse value

search column = search column - 1

END

END

ELSE

BEGIN

As for case where layer number = 1 but read synapse values from previous size layer

END

Comment search to the right

As for above but with the following differences:

column start = neuron column + 1

group counter = 2

position 2 = column counter - column start

increment column counter until the end of the layer is reached

END

CASE Vertical

BEGIN

As for Horizontal but

IF group counter = 1 **THEN**

BEGIN

row start = neuron row - 1

position 1 = row start – row counter

decrement row counter until the end of the layer is reached

END

IF group counter = 2 **THEN**

BEGIN

row start = neuron row + 1

position 2 = row counter – row start

increment row counter until the end of the layer is reached

END

END

Pseudocode A5.2 **Size layer dendrite searching**

The above algorithm is contained in a function that returns the array of synapse input values and the values for position 1 and position 2. A position value represents the distance from the neuron body to the maximum input value. There is a value for each dendrite group.

The output from a size layer neuron is calculated using the following algorithm:

```
Get synapse inputs for neuron using Pseudocode A5.2
and sum = 0
synapse sum = 0
abs error sum = 0
FOR all dendrite groups DO
BEGIN
    FOR all dendrites in a group DO
    BEGIN
        determine input for dendrite
        determine weight for dendrite
        max input for dendrite = max(max input for dendrite, input)
        abs error sum = abs error sum + |input - weight|
        and sum = and sum + min(input, weight)
        synapse sum = synapse sum + weight
    END
END
```

Use these values to calculate the neuron output

net = 1 - abs error sum / (dendrites per group * dendrite groups)

and net = and sum / synapse sum

Calculate the compensation terms to bias the neuron

centre offset = |position 1 - position 2|

IF (centre offset > 1) **THEN**

BEGIN

net = net / centre offset

and net = and net / centre offset

END

and net = and net * contribution from retina

net = net * (1 - contribution from previous layer)

and net = and net * (1 - contribution from previous layer)

and neuron output = activation function(and net)

neuron output = activation function(net)

tired output = tire output(and neuron output)

Pseudocode A5.3 Size neuron output algorithm

CORNER LAYER

The corner layer dendrites are searched using the following algorithm:

Assign neuron search direction from neuron number

IF neuron search direction = LEFT BOTTOM

OR

neuron search direction = LEFT TOP

```
THEN
BEGIN
        Comment search to the left
        synapses initialised = FALSE
        largest found = FALSE
        row start = neuron row + 1
        column start  = neuron column
        previous max synapse value = -1
        group counter = 1
        WHILE (column counter >= 1
                AND
                NOT largest found) DO
        BEGIN
                row = neuron row
                column counter = column start
                max synapse value = -1
                FOR dendrite counter = 1 TO number of dendrites
                BEGIN
                        determine synapse array index for dendrite counter
                        IF NOT synapses initialise THEN
                        BEGIN
                                synapse array [index] = -1
                        END
                        read input value from last size layer at position row, column
                        temp synapse values [dendrite counter - 1] = input value
                        max synapse value = Max(max synapse value, input value)
```

END

synapses initialised = TRUE

IF (max synapse value > previous max synapse value) **THEN**

BEGIN

IF (max synapse value = 1) **THEN**

BEGIN

largest found = TRUE

END

END

copy values from temp synapse values to synapse array

column position = column counter

previous max synapse value = max synapse value

decrement column counter

END

END

IF neuron search direction = RIGHT BOTTOM

OR

neuron search direction = RIGHT TOP

THEN

BEGIN

Comment search to the right

As for *search to the left* but with the following differences:

column start = neuron column + 2

group counter = 2

increment column counter until the end of the layer is reached

END

IF neuron search direction = RIGHT BOTTOM

OR

neuron search direction = LEFT BOTTOM

THEN

BEGIN

Comment search to the bottom

as for *search to the left* but with the following differences:

row start = neuron row

column start = neuron column + 1

group counter = 2

row position = row counter

decrement row counter until the end of the layer is reached

END

IF neuron search direction = RIGHT TOP

OR

neuron search direction = LEFT TOP

THEN

BEGIN

Comment search to the bottom

as for *search to the bottom* but with the following differences:

row start = neuron row + 2

increment row counter until the end of the layer is reached

END

Pseudocode A5.4 **Corner layer dendrite search**

The above algorithm is contained in a function that returns the array of synapse input values and the values for row position and column position. A position value represents the position of the maximum input value. There is a value for each dendrite group.

The output from a corner layer neuron is calculated using the following algorithm:

```
Get synapse inputs for neuron using Pseudocode A5.4

and sum = 0

synapse sum = 0

abs error sum = 0

FOR all dendrite groups DO
BEGIN
    FOR all dendrites in a group DO
    BEGIN
        determine input for dendrite

        determine weight for dendrite

        abs error sum = abs error sum + |input - weight|

        and sum = and sum + min(input, weight)

        synapse sum = synapse sum + weight

    END

END

Use these values to calculate the neuron output

    net = 1 - abs error sum / (dendrites per group * dendrite groups)
```

and net = and sum / synapse sum

Calculate the compensation terms to bias the neuron

net = net * contribution from retina

and net = and net * contribution from retina

and neuron output = activation function(and net)

neuron output = activation function(net)

tired output = tire output(and neuron output)

Pseudocode A5.5 **Corner neuron output algorithm**

SIZE 2 LAYER

The size 2 layer dendrites are searched using the following algorithm:

Assign neuron search direction from neuron number

SWITCH neuron search direction

CASE Horizontal

BEGIN

synapses initialised = FALSE

largest found = FALSE

column start = neuron column - 1

previous max synapse value = -1

group counter = 1

WHILE (column counter >= 1

AND

NOT largest found) **DO**

BEGIN

row = neuron row

```
column counter = column start

max synapse value = -1

FOR dendrite counter = 1 TO number of dendrites

BEGIN

        determine synapse array index for dendrite counter

        IF NOT synapses initialise THEN

        BEGIN

                synapse array [index] = -1

        END

        read input value from last corner layer at position row, column

        temp synapse values [dendrite counter - 1] = input value

        max synapse value = Max(max synapse value, input value)


END

synapses initialised = TRUE

IF (max synapse value > previous max synapse value) THEN

BEGIN

        IF (max synapse value = 1) THEN

        BEGIN

                largest found = TRUE

        END

END

copy values from temp synapse values to synapse array

position 1 = column start - column counter

previous max synapse value = max synapse value

search column = search column - 1
```

END

Comment search to the right

As for above but with the following differences:

column start = neuron column + 1

group counter = 2

position 2 = column counter - column start

increment column counter until the end of the layer is reached

END

CASE Vertical

BEGIN

As for Horizontal but

IF group counter = 1 **THEN**

BEGIN

row start = neuron row - 1

position 1 = row start – row counter

decrement row counter until the end of the layer is reached

END

IF group counter = 2 **THEN**

BEGIN

row start = neuron row + 1

position 2 = row counter – row start

increment row counter until the end of the layer is reached

END

END

Pseudocode A5.6 Size 2 layer dendrite search

The above algorithm is contained in a function that returns the array of synapse input values and the values for position 1 and position 2. A position value represents the distance from the neuron body to the maximum input value. There is a value for each dendrite group.

The output from a size 2 layer neuron is calculated using the following algorithm:

Get synapse inputs for neuron using Pseudocode A5.6

and sum = 0

synapse sum = 0

abs error sum = 0

FOR all dendrite groups **DO**

BEGIN

FOR all dendrites in a group **DO**

BEGIN

determine input for dendrite

determine weight for dendrite

abs error sum = abs error sum + |input - weight|

and sum = and sum + min(input, weight)

synapse sum = synapse sum + weight

END

END

Use these values to calculate the neuron output

net = 1 - abs error sum / (dendrites per group * dendrite groups)

and net = and sum / synapse sum

Calculate the compensation terms to bias the neuron

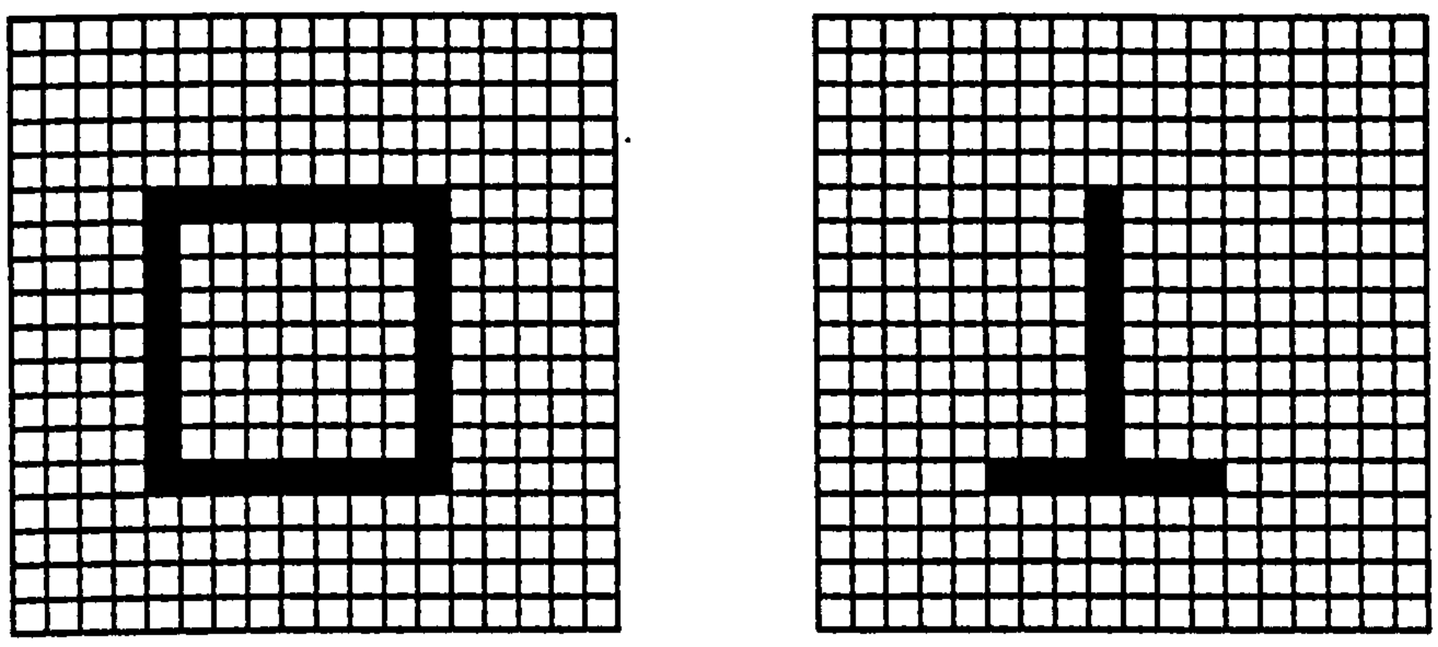
```
centre offset = |position 1 - position 2|

IF (centre offset > 1) THEN

BEGIN

        net = net / centre offset

        and net = and net / centre offset

END

        net = net * contribution from retina

        and net = and net * contribution from retina

        and neuron output = activation function(and net)

        neuron output = activation function(net)

        tired output = tire output(and neuron output)
```

Pseudocode A5.7 **Size 2 neuron output algorithm**

APPENDIX 6 – SCALE INVARIANT NETWORK

This appendix demonstrates the behaviour of the scale invariant network. A triple layer network was used to classify the images 0 to 9 and decompose the images into their structural components.

The network was tested using 18 rows × 18 columns images that are shown in Figure A6.1.



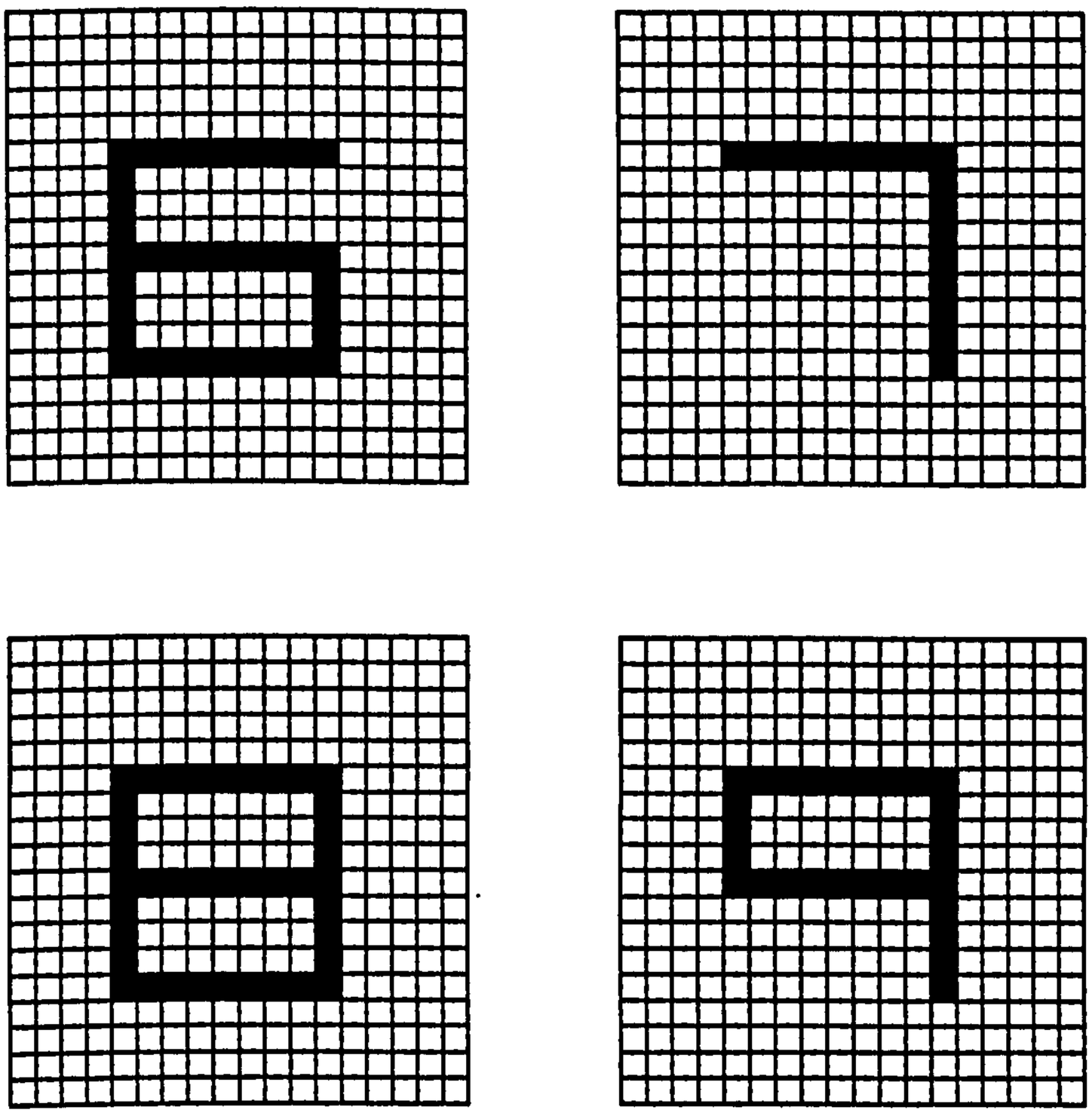


Figure A6.1 Images used to train the classifier

A6.1 IMAGE CLASSIFICATION

The network was classified using a four layer network constructed of the following layers:

- Primitive layer
- Size layer
- Corner layer
- Size 2 layer

The primitive layer is not taught but is constructed in a manner to identify the primitives in the image so only last three layers are taught. The fixed synapse weights were set to the values specified in Chapter 3.

The primitive layer consisted of 17 features each with a receptive field of 7 rows × 7 columns, as shown in Figure A6.2.

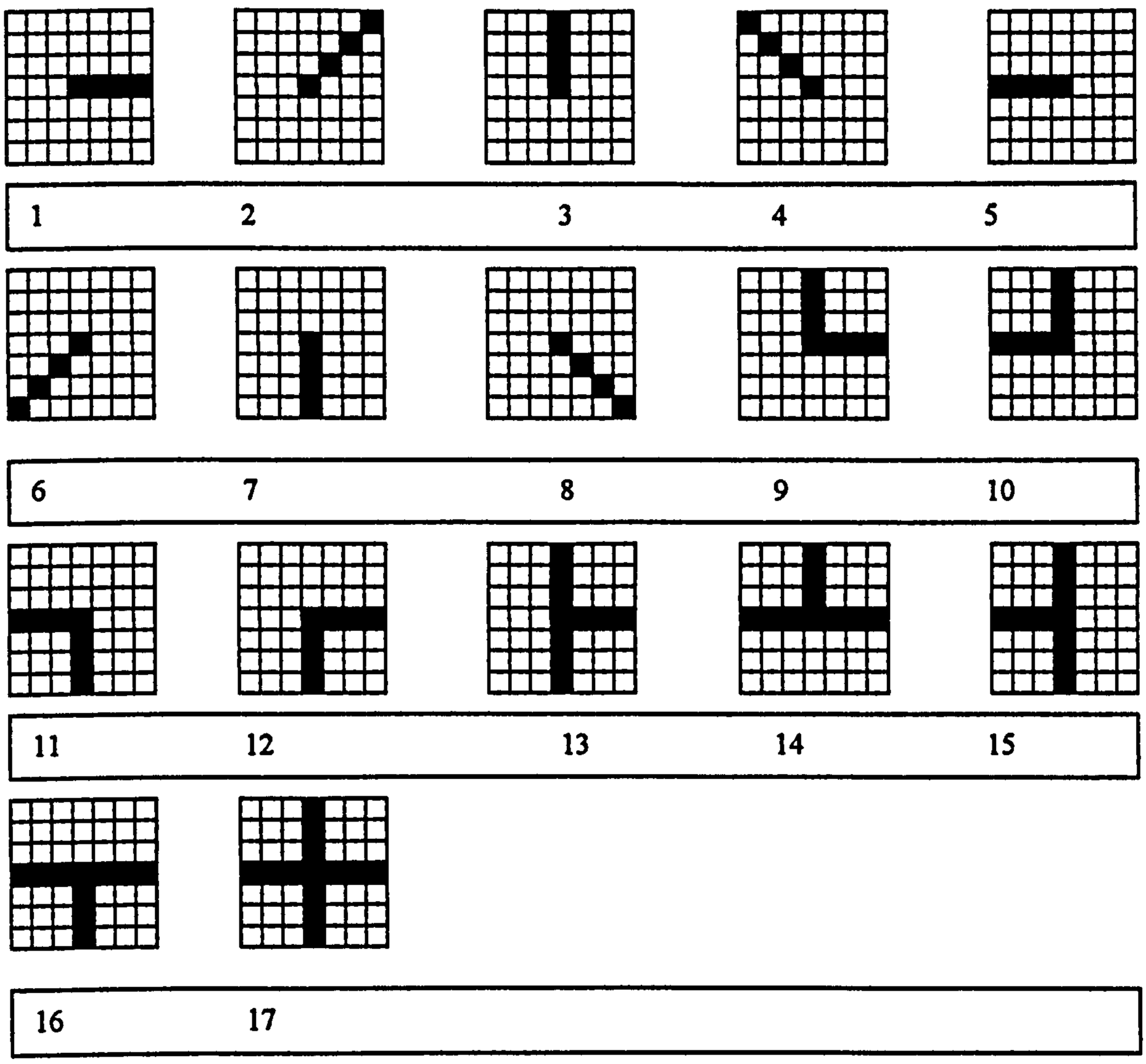


Figure A6.2 Primitives used

The primitives used can identify the following types of features:

- 4 horizontal or vertical edges
- 4 edges at 45° to horizontal or vertical
- 4 tee junctions
- 1 cross

The non-horizontal or vertical edge detectors were used because it was originally planned to extend the system to classify features that were not horizontal or vertical. It was decided

to leave these features in the system to determine whether they would have any effect on the performance of the system.

Teaching the network was very time consuming. For this reason, it was decided to partition the teaching into the teaching of each layer. One layer was taught and the resulting synapse values were written to a file. The next layer is taught by reloading the synapses values and teaching the next layer. The network was taught in three parts: teaching the size layer, teaching the corner layer and finally teaching the size 2 layer.

The parameters for each layer are shown in Table A6.1

Layer	Tiredness reduction	Tiredness enhancement	Atrophy	Tolerance	Learning rate	Teaching presentations
Primitive	0.001	0.9	0.25	0.25	0.5	0
Size	0.01	0.8	0.25	0.5	0.5	40
Corner	0.03	0.97	0.25	0.02	0.5	25
Size 2	0.01	0.8	0.0	0.05	0.5	13

Table A6.1 Size network layer parameters

The maximum output from the primitive layer is shown in Results A6.1.

```
Layer 1 image number 1
0.319 0.473 0.217 0.217 0.217 0.217 0.217 0.217 0.217 0.473 0.319 0.319
0.237 0.722 0.165 0.217 0.217 0.217 0.217 0.217 0.165 0.722 0.237 0.319
0.722 1.000 0.722 0.722 0.493 0.669 0.493 0.722 0.722 1.000 0.722 0.473
0.165 0.722 0.112 0.148 0.148 0.217 0.148 0.148 0.112 0.722 0.148 0.237
0.217 0.722 0.148 0.148 0.148 0.217 0.165 0.148 0.148 0.722 0.217 0.217
0.217 0.493 0.148 0.148 0.148 0.217 0.165 0.148 0.148 0.493 0.217 0.217
0.217 0.669 0.217 0.217 0.217 0.342 0.237 0.217 0.217 0.669 0.217 0.217
0.217 0.493 0.148 0.165 0.165 0.237 0.148 0.148 0.148 0.493 0.217 0.217
0.217 0.722 0.148 0.148 0.148 0.217 0.148 0.148 0.148 0.722 0.217 0.217
0.165 0.722 0.112 0.148 0.148 0.217 0.148 0.148 0.114 0.722 0.148 0.237
0.722 1.000 0.722 0.722 0.493 0.669 0.493 0.722 0.722 1.000 0.722 0.473
0.237 0.722 0.148 0.217 0.217 0.217 0.217 0.217 0.148 0.722 0.237 0.319
```

Layer 1 image number 2
0.342 0.342 0.342 0.342 0.319 0.722 0.342 0.319 0.319 0.342 0.342 0.342
0.342 0.342 0.342 0.319 0.342 1.000 0.319 0.319 0.319 0.342 0.342 0.342
0.342 0.342 0.342 0.319 0.319 1.000 0.319 0.319 0.319 0.342 0.342 0.342
0.342 0.342 0.342 0.237 0.217 1.000 0.217 0.237 0.319 0.342 0.342 0.342
0.342 0.342 0.342 0.217 0.217 0.722 0.217 0.217 0.319 0.342 0.342 0.342
0.342 0.342 0.237 0.217 0.217 0.669 0.217 0.217 0.217 0.342 0.342 0.342
0.342 0.342 0.237 0.217 0.217 0.669 0.217 0.217 0.217 0.342 0.342 0.342
0.342 0.342 0.217 0.237 0.165 0.473 0.148 0.217 0.217 0.319 0.319 0.473
0.342 0.319 0.217 0.148 0.148 0.722 0.148 0.148 0.217 0.319 0.319 0.342
0.319 0.342 0.217 0.148 0.101 0.473 0.101 0.165 0.217 0.319 0.342 0.342
0.722 1.000 1.000 0.722 0.473 1.000 0.473 0.722 1.000 1.000 0.722 0.473
0.342 0.319 0.319 0.217 0.148 0.473 0.148 0.217 0.319 0.319 0.342 0.342

Layer 1 image number 3
0.319 0.319 0.217 0.217 0.217 0.217 0.217 0.217 0.217 0.473 0.319 0.319
0.342 0.319 0.217 0.217 0.217 0.217 0.217 0.217 0.165 0.722 0.237 0.319
1.000 1.000 1.000 0.722 0.669 0.669 0.493 0.722 0.722 1.000 0.722 0.473
0.237 0.217 0.148 0.148 0.148 0.148 0.101 0.101 0.081 0.473 0.148 0.319
0.319 0.319 0.148 0.148 0.148 0.148 0.101 0.101 0.101 0.319 0.217 0.217
0.217 0.473 0.114 0.148 0.148 0.148 0.101 0.101 0.081 0.473 0.148 0.319
0.722 1.000 0.722 0.722 0.493 0.669 0.493 0.722 0.722 1.000 0.722 0.473
0.165 0.473 0.081 0.101 0.101 0.148 0.148 0.148 0.101 0.473 0.217 0.319
0.217 0.319 0.101 0.101 0.101 0.148 0.148 0.148 0.148 0.319 0.319 0.319
0.148 0.473 0.081 0.101 0.101 0.148 0.148 0.148 0.165 0.217 0.217 0.319
0.722 1.000 0.722 0.722 0.493 0.669 0.669 0.722 1.000 1.000 1.000 0.722
0.237 0.722 0.148 0.217 0.217 0.217 0.217 0.217 0.217 0.319 0.319 0.342

Layer 1 image number 4
0.319 0.319 0.217 0.217 0.217 0.217 0.217 0.217 0.217 0.473 0.319 0.319
0.342 0.319 0.217 0.217 0.217 0.217 0.217 0.217 0.165 0.722 0.237 0.319
1.000 1.000 1.000 0.722 0.669 0.669 0.493 0.722 0.722 1.000 0.722 0.473
0.237 0.217 0.148 0.148 0.148 0.148 0.101 0.101 0.081 0.473 0.148 0.319
0.319 0.217 0.148 0.148 0.148 0.148 0.101 0.101 0.101 0.473 0.217 0.217
0.237 0.217 0.148 0.148 0.148 0.148 0.101 0.101 0.068 0.370 0.148 0.217
1.000 1.000 1.000 0.722 0.669 0.669 0.473 0.722 0.473 1.000 0.473 0.319
0.237 0.217 0.148 0.148 0.148 0.148 0.101 0.101 0.068 0.370 0.148 0.217
0.319 0.217 0.148 0.148 0.148 0.148 0.101 0.101 0.101 0.473 0.217 0.217
0.237 0.217 0.148 0.148 0.148 0.148 0.101 0.101 0.081 0.473 0.148 0.319
1.000 1.000 1.000 0.722 0.669 0.669 0.493 0.722 0.722 1.000 0.722 0.473
0.319 0.319 0.217 0.217 0.217 0.217 0.217 0.217 0.148 0.722 0.237 0.319

Layer 1 image number 5
0.319 1.000 0.319 0.319 0.319 0.342 0.342 0.342 0.342 0.342 0.342 0.342
0.217 1.000 0.217 0.237 0.319 0.342 0.342 0.342 0.342 0.342 0.342 0.342
0.217 0.722 0.217 0.237 0.319 0.342 0.473 0.342 0.342 0.473 0.342 0.342
0.217 0.669 0.217 0.217 0.217 0.319 0.722 0.342 0.319 0.319 0.342 0.342
0.217 0.669 0.217 0.217 0.217 0.342 1.000 0.319 0.319 0.319 0.342 0.342
0.217 0.493 0.148 0.148 0.165 0.217 0.722 0.217 0.319 0.237 0.319 0.319
0.217 0.722 0.148 0.114 0.148 0.148 0.473 0.148 0.217 0.319 0.319 0.319
0.165 0.722 0.112 0.114 0.101 0.068 0.319 0.078 0.148 0.217 0.319 0.342
0.722 1.000 0.722 0.319 0.319 0.217 1.000 0.319 0.473 0.722 1.000 0.722
0.237 0.722 0.148 0.148 0.148 0.078 0.319 0.068 0.148 0.217 0.319 0.342
0.319 0.473 0.237 0.217 0.217 0.148 0.473 0.148 0.237 0.319 0.319 0.319
0.319 0.319 0.319 0.237 0.217 0.217 0.722 0.217 0.319 0.237 0.319 0.319

Layer 1 image number 6
0.319 0.473 0.217 0.217 0.217 0.217 0.217 0.217 0.217 0.319 0.319 0.319
0.237 0.722 0.165 0.217 0.217 0.217 0.217 0.217 0.237 0.319 0.319 0.342
0.722 1.000 0.722 0.722 0.493 0.669 0.669 0.722 1.000 1.000 1.000 0.722
0.165 0.473 0.081 0.101 0.101 0.148 0.148 0.148 0.148 0.217 0.217 0.319
0.217 0.319 0.101 0.101 0.101 0.148 0.148 0.148 0.148 0.319 0.319 0.319
0.148 0.473 0.081 0.101 0.101 0.148 0.148 0.148 0.114 0.473 0.217 0.319
0.722 1.000 0.722 0.722 0.493 0.669 0.493 0.722 0.722 1.000 0.722 0.473
0.237 0.473 0.114 0.148 0.148 0.148 0.101 0.101 0.081 0.473 0.148 0.319
0.319 0.319 0.148 0.148 0.148 0.148 0.101 0.101 0.101 0.319 0.217 0.217
0.237 0.217 0.148 0.148 0.148 0.148 0.101 0.101 0.081 0.473 0.148 0.319
1.000 1.000 1.000 0.722 0.669 0.669 0.493 0.722 0.722 1.000 0.722 0.473
0.319 0.319 0.217 0.217 0.217 0.217 0.217 0.217 0.148 0.722 0.237 0.319

```
Layer 1 image number 7
0.319 0.473 0.217 0.217 0.217 0.217 0.217 0.217 0.217 0.319 0.319 0.319
0.237 0.722 0.165 0.217 0.217 0.217 0.217 0.217 0.237 0.319 0.319 0.342
0.722 1.000 0.722 0.722 0.493 0.669 0.669 0.722 1.000 1.000 1.000 0.722
0.165 0.473 0.081 0.101 0.101 0.148 0.148 0.148 0.148 0.217 0.217 0.319
0.217 0.473 0.101 0.101 0.101 0.148 0.148 0.148 0.148 0.319 0.319 0.319
0.148 0.370 0.068 0.101 0.101 0.148 0.148 0.148 0.114 0.473 0.217 0.319
0.473 1.000 0.473 0.722 0.473 0.669 0.493 0.722 0.722 1.000 0.722 0.473
0.165 0.370 0.068 0.101 0.101 0.148 0.101 0.101 0.081 0.473 0.148 0.319
0.217 0.473 0.101 0.101 0.101 0.148 0.101 0.101 0.101 0.319 0.217 0.217
0.148 0.473 0.081 0.101 0.101 0.148 0.101 0.101 0.081 0.473 0.148 0.319
0.722 1.000 0.722 0.722 0.493 0.669 0.493 0.722 0.722 1.000 0.722 0.473
0.237 0.722 0.148 0.217 0.217 0.217 0.217 0.217 0.148 0.722 0.237 0.319

Layer 1 image number 8
0.319 0.319 0.217 0.217 0.217 0.217 0.217 0.217 0.217 0.473 0.319 0.319
0.342 0.319 0.217 0.217 0.217 0.217 0.217 0.217 0.165 0.722 0.237 0.319
1.000 1.000 1.000 0.722 0.669 0.669 0.493 0.722 0.722 1.000 0.722 0.473
0.319 0.319 0.217 0.217 0.217 0.217 0.148 0.148 0.112 0.722 0.148 0.237
0.319 0.319 0.237 0.217 0.217 0.217 0.165 0.148 0.148 0.722 0.217 0.217
0.319 0.319 0.319 0.319 0.217 0.217 0.165 0.148 0.148 0.493 0.217 0.217
0.342 0.342 0.342 0.342 0.342 0.342 0.237 0.217 0.217 0.669 0.217 0.217
0.342 0.342 0.342 0.342 0.342 0.342 0.237 0.217 0.217 0.669 0.217 0.217
0.342 0.342 0.342 0.342 0.342 0.342 0.319 0.237 0.217 0.722 0.217 0.217
0.342 0.342 0.342 0.342 0.342 0.342 0.319 0.217 0.237 1.000 0.217 0.237
0.342 0.342 0.342 0.342 0.342 0.342 0.319 0.319 0.319 1.000 0.319 0.319
0.342 0.342 0.342 0.342 0.342 0.342 0.319 0.319 0.319 1.000 0.319 0.319

Layer 1 image number 9
0.319 0.473 0.217 0.217 0.217 0.217 0.217 0.217 0.217 0.473 0.319 0.319
0.237 0.722 0.165 0.217 0.217 0.217 0.217 0.217 0.165 0.722 0.237 0.319
0.722 1.000 0.722 0.722 0.493 0.669 0.493 0.722 0.722 1.000 0.722 0.473
0.165 0.473 0.081 0.101 0.101 0.148 0.101 0.101 0.081 0.473 0.148 0.319
0.217 0.473 0.101 0.101 0.101 0.148 0.101 0.101 0.101 0.473 0.217 0.217
0.148 0.370 0.068 0.101 0.101 0.148 0.101 0.101 0.068 0.370 0.148 0.217
0.473 1.000 0.473 0.722 0.473 0.669 0.473 0.722 0.473 1.000 0.473 0.319
0.165 0.370 0.068 0.101 0.101 0.148 0.101 0.101 0.068 0.370 0.148 0.217
0.217 0.473 0.101 0.101 0.101 0.148 0.101 0.101 0.101 0.473 0.217 0.217
0.148 0.473 0.081 0.101 0.101 0.148 0.101 0.101 0.081 0.473 0.148 0.319
0.722 1.000 0.722 0.722 0.493 0.669 0.493 0.722 0.722 1.000 0.722 0.473
0.237 0.722 0.148 0.217 0.217 0.217 0.217 0.217 0.148 0.722 0.237 0.319

Layer 1 image number 10
0.319 0.473 0.217 0.217 0.217 0.217 0.217 0.217 0.217 0.473 0.319 0.319
0.237 0.722 0.165 0.217 0.217 0.217 0.217 0.217 0.165 0.722 0.237 0.319
0.722 1.000 0.722 0.722 0.493 0.669 0.493 0.722 0.722 1.000 0.722 0.473
0.165 0.473 0.081 0.101 0.101 0.148 0.101 0.101 0.081 0.473 0.148 0.319
0.217 0.319 0.101 0.101 0.101 0.148 0.101 0.101 0.101 0.473 0.217 0.217
0.148 0.473 0.081 0.101 0.101 0.148 0.101 0.101 0.068 0.370 0.148 0.217
0.722 1.000 0.722 0.722 0.493 0.669 0.473 0.722 0.473 1.000 0.473 0.319
0.237 0.722 0.148 0.217 0.217 0.217 0.148 0.148 0.101 0.493 0.148 0.217
0.319 0.473 0.237 0.217 0.217 0.217 0.217 0.165 0.148 0.473 0.217 0.217
0.319 0.319 0.319 0.319 0.217 0.217 0.217 0.217 0.165 0.722 0.217 0.319
0.342 0.342 0.342 0.342 0.342 0.342 0.319 0.319 0.319 1.000 0.319 0.319
0.342 0.342 0.342 0.342 0.342 0.342 0.319 0.319 0.319 1.000 0.319 0.319
```

Results A6.1 Maximum outputs in primitive layer

The maximum output values for the size layer are shown in Results A6.2.

```
Size layer 1 image number 1
0.658 0.060 0.085 0.147 0.574 0.574 0.147 0.085 0.658 0.047
0.000 0.092 0.142 0.312 1.000 0.312 0.142 0.092 0.000 0.602
0.092 0.089 0.090 0.149 0.583 0.583 0.149 0.088 0.092 0.086
0.142 0.153 0.155 0.156 0.609 0.609 0.153 0.153 0.142 0.148
0.312 0.613 0.623 0.631 0.639 0.630 0.610 0.607 0.312 0.576
1.000 0.613 0.623 0.631 0.642 0.642 0.610 0.607 1.000 0.576
0.312 0.153 0.155 0.156 0.623 0.623 0.155 0.153 0.312 0.148
0.142 0.089 0.090 0.152 0.604 0.604 0.152 0.088 0.142 0.086
0.092 0.063 0.089 0.153 0.613 0.613 0.153 0.089 0.092 0.060
0.000 0.092 0.142 0.312 1.000 0.312 0.142 0.092 0.000 0.645

Size layer 1 image number 2
0.155 0.620 0.622 0.155 0.000 0.110 0.236 0.590 0.236 0.110
0.153 0.608 0.637 0.157 0.000 0.109 0.233 0.578 0.233 0.109
0.155 0.620 0.639 0.157 0.000 0.110 0.236 0.590 0.236 0.110
0.243 0.246 0.156 0.153 0.613 0.613 0.156 0.239 0.235 0.148
0.617 0.627 0.632 0.612 0.598 0.598 0.632 0.599 0.587 0.581
0.243 0.245 0.632 0.612 1.000 0.598 0.632 0.238 0.235 0.581
0.113 0.114 0.156 0.154 1.000 0.619 0.156 0.111 0.110 0.148
0.074 0.075 0.091 0.154 0.614 0.614 0.154 0.089 0.072 0.086
0.055 0.061 0.087 0.150 0.588 0.588 0.150 0.087 0.061 0.061
0.000 0.000 1.000 1.000 0.000 0.720 0.720 0.000 0.000 0.592

Size layer 1 image number 3
0.641 0.640 0.088 0.151 0.600 0.600 0.151 0.088 0.658 0.053
0.000 0.000 0.111 0.194 1.000 1.000 0.194 0.111 0.000 0.602
0.235 0.063 0.089 0.153 0.612 0.612 0.153 0.112 0.312 0.109
0.589 0.087 0.154 0.156 0.621 0.621 0.155 0.241 1.000 0.234
0.235 0.150 0.618 0.632 0.639 0.632 0.620 0.609 0.312 0.585
0.000 0.593 0.618 0.632 1.000 0.632 0.620 0.241 0.000 0.645
0.312 0.593 0.154 0.156 0.582 0.582 0.155 0.112 0.240 0.109
1.000 0.150 0.090 0.154 0.620 0.620 0.154 0.090 0.607 0.072
0.312 0.087 0.088 0.151 0.597 0.597 0.151 0.088 0.240 0.053
0.000 0.111 0.194 1.000 1.000 0.194 0.111 0.000 0.000 0.000

Size layer 1 image number 4
0.641 0.640 0.088 0.151 0.600 0.600 0.151 0.088 0.658 0.048
0.000 0.000 0.111 0.194 1.000 1.000 0.194 0.111 0.000 0.602
0.238 0.241 0.090 0.153 0.612 0.612 0.153 0.089 0.312 0.086
0.599 0.608 0.155 0.157 0.628 0.628 0.156 0.153 1.000 0.148
0.238 0.241 0.625 0.639 0.639 0.625 0.610 0.607 0.312 0.576
0.000 0.000 0.625 0.639 1.000 1.000 0.610 0.607 0.000 0.587
0.238 0.241 0.155 0.157 0.612 0.612 0.153 0.153 0.312 0.148
0.599 0.608 0.091 0.157 0.637 0.637 0.157 0.091 1.000 0.086
0.238 0.241 0.089 0.153 0.612 0.612 0.153 0.089 0.312 0.060
0.000 0.000 0.111 0.194 1.000 1.000 0.194 0.111 0.000 0.645

Size layer 1 image number 5
0.000 0.053 0.072 0.110 0.236 0.590 0.236 0.110 0.072 0.072
0.111 0.063 0.088 0.152 0.605 0.605 0.152 0.088 0.062 0.110
0.194 0.089 0.091 0.156 0.632 0.632 0.156 0.091 0.089 0.235
1.000 0.245 0.626 0.245 0.149 0.000 0.150 0.590 0.590 0.587
1.000 0.619 0.628 0.616 0.642 0.642 0.577 0.607 0.611 0.235
0.194 0.619 0.628 0.616 0.618 1.000 0.577 0.607 0.611 0.110
0.111 0.154 0.156 0.154 0.588 0.588 0.150 0.153 0.153 0.072
0.000 0.194 1.000 1.000 0.194 0.000 0.260 0.693 0.260 0.000
0.616 0.063 0.087 0.150 0.591 0.677 0.150 0.087 0.063 0.623
0.595 0.064 0.091 0.157 0.639 0.677 0.157 0.091 0.065 0.619
```

```
Size layer 1 image number 6
0.658 0.060 0.084 0.145 0.563 0.563 0.145 0.617 0.616 0.592
0.000 0.100 0.174 0.778 0.778 0.174 0.100 0.000 0.000 0.000
0.312 0.115 0.090 0.147 0.573 0.573 0.147 0.085 0.235 0.062
1.000 0.247 0.155 0.157 0.608 0.608 0.154 0.089 0.589 0.087
0.312 0.632 0.622 0.638 0.639 0.632 0.615 0.150 0.235 0.151
0.000 0.247 0.622 0.638 1.000 0.632 0.615 0.592 0.000 0.602
0.235 0.115 0.155 0.157 0.610 0.610 0.154 0.592 0.312 0.594
0.589 0.075 0.090 0.155 0.621 0.621 0.155 0.150 1.000 0.151
0.235 0.063 0.089 0.153 0.612 0.612 0.153 0.089 0.312 0.087
0.000 0.000 0.111 0.194 1.000 1.000 0.194 0.111 0.000 0.645

Size layer 1 image number 7
0.658 0.060 0.084 0.145 0.563 0.563 0.145 0.617 0.616 0.592
0.000 0.100 0.174 0.778 0.778 0.174 0.100 0.000 0.000 0.000
0.312 0.089 0.090 0.147 0.573 0.573 0.147 0.085 0.235 0.062
1.000 0.153 0.155 0.156 0.602 0.602 0.154 0.088 0.589 0.087
0.312 0.613 0.623 0.631 0.639 0.632 0.615 0.150 0.235 0.151
0.000 0.613 0.623 0.631 1.000 0.632 0.615 0.592 0.000 0.602
0.312 0.153 0.155 0.156 0.590 0.590 0.154 0.592 0.312 0.594
1.000 0.089 0.090 0.152 0.605 0.605 0.152 0.150 1.000 0.151
0.312 0.063 0.088 0.152 0.606 0.606 0.152 0.088 0.312 0.087
0.000 0.092 0.142 0.312 1.000 0.312 0.142 0.092 0.000 0.645

Size layer 1 image number 8
0.641 0.640 0.088 0.151 0.600 0.600 0.151 0.088 0.658 0.048
0.000 0.000 0.111 0.194 1.000 1.000 0.194 0.111 0.000 0.602
0.061 0.062 0.089 0.153 0.609 0.609 0.153 0.089 0.111 0.087
0.087 0.087 0.151 0.157 0.639 0.639 0.158 0.153 0.194 0.150
0.150 0.149 0.599 0.632 0.646 0.646 0.641 0.610 1.000 0.591
0.591 0.587 0.599 0.632 0.635 0.642 0.641 0.610 1.000 0.591
0.591 0.587 0.151 0.157 0.635 0.635 0.158 0.153 0.194 0.150
0.150 0.149 0.090 0.155 0.626 0.626 0.155 0.090 0.111 0.087
0.087 0.090 0.156 0.628 0.628 0.156 0.090 0.064 0.000 0.640
0.064 0.091 0.158 0.642 0.642 0.158 0.091 0.064 0.000 0.641

Size layer 1 image number 9
0.658 0.060 0.085 0.147 0.574 0.574 0.147 0.085 0.658 0.047
0.000 0.092 0.142 0.312 1.000 0.312 0.142 0.092 0.000 0.602
0.312 0.089 0.090 0.149 0.587 0.587 0.149 0.088 0.312 0.086
1.000 0.153 0.155 0.156 0.608 0.608 0.153 0.153 1.000 0.148
0.312 0.613 0.623 0.631 0.639 0.625 0.610 0.607 0.312 0.576
0.000 0.613 0.623 0.631 0.766 0.625 0.610 0.607 0.000 0.587
0.312 0.153 0.155 0.156 0.590 0.590 0.153 0.153 0.312 0.148
1.000 0.089 0.090 0.153 0.610 0.610 0.153 0.089 1.000 0.086
0.312 0.063 0.088 0.152 0.606 0.606 0.152 0.088 0.312 0.060
0.000 0.092 0.142 0.312 1.000 0.312 0.142 0.092 0.000 0.645

Size layer 1 image number 10
0.658 0.060 0.085 0.147 0.574 0.574 0.147 0.085 0.658 0.047
0.000 0.092 0.142 0.312 1.000 0.312 0.142 0.092 0.000 0.602
0.312 0.089 0.087 0.149 0.587 0.587 0.149 0.089 0.312 0.087
1.000 0.153 0.151 0.154 0.608 0.608 0.158 0.153 1.000 0.150
0.312 0.609 0.594 0.614 0.632 0.642 0.641 0.610 0.312 0.591
0.000 0.609 0.594 0.614 1.000 0.642 0.641 0.610 0.000 0.591
0.148 0.153 0.151 0.154 0.608 0.608 0.158 0.153 0.312 0.150
0.580 0.089 0.091 0.157 0.639 0.639 0.157 0.091 1.000 0.087
0.580 0.064 0.091 0.156 0.629 0.629 0.156 0.090 0.312 0.061
0.148 0.091 0.158 0.642 0.642 0.158 0.091 0.064 0.000 0.641
```

Results A6.2 Maximum outputs in size layer

The maximum results from the corner layer are shown in Results A6.3.

```
Corn layer 1 image number 1
0.287 0.396 0.436 0.508 0.348 0.296 0.286 0.148
0.305 0.393 0.423 0.580 0.499 0.402 0.384 0.280
0.312 0.412 0.449 0.638 0.687 0.548 0.525 0.350
0.333 0.509 0.554 0.860 1.000 0.706 0.672 0.407
0.371 0.486 0.522 0.842 0.616 0.496 0.476 0.517
0.333 0.431 0.463 0.637 0.545 0.440 0.422 0.426
0.312 0.402 0.432 0.593 0.509 0.410 0.393 0.389
0.235 0.312 0.334 0.440 0.334 0.312 0.305 0.287

Corn layer 1 image number 2
0.065 0.154 0.243 0.064 0.247 0.247 0.248 0.155
0.066 0.156 0.247 0.064 0.249 0.249 0.250 0.157
0.067 0.140 0.219 0.074 0.219 0.219 0.222 0.140
0.067 0.161 0.254 0.080 0.248 0.248 0.224 0.141
0.343 0.609 1.000 0.081 0.882 0.883 0.627 0.463
0.350 0.613 1.000 0.212 0.892 0.894 0.639 0.470
0.053 0.131 0.204 0.214 0.255 0.249 0.212 0.134
0.059 0.156 0.246 0.188 0.249 0.246 0.198 0.127

Corn layer 1 image number 3
0.190 0.274 0.321 0.314 0.339 0.336 0.336 0.146
0.118 0.187 0.216 0.264 0.264 0.221 0.220 0.065
0.364 0.479 0.610 1.000 1.000 0.617 0.596 0.338
0.253 0.340 0.388 0.470 0.469 0.415 0.352 0.398
0.291 0.522 0.622 1.000 0.414 0.356 0.350 0.403
0.337 0.517 0.600 0.886 0.780 0.512 0.500 0.618
0.375 0.496 0.623 1.000 0.739 0.495 0.478 0.509
0.257 0.346 0.394 0.558 0.475 0.335 0.313 0.313

Corn layer 1 image number 4
0.194 0.266 0.274 0.315 0.269 0.294 0.286 0.147
0.138 0.184 0.189 0.265 0.264 0.189 0.184 0.066
0.460 0.483 0.500 1.000 1.000 0.500 0.483 0.153
0.215 0.255 0.306 0.411 0.380 0.369 0.360 0.180
0.194 0.329 0.338 0.413 0.399 0.359 0.350 0.173
0.138 0.184 0.189 0.274 0.272 0.189 0.184 0.069
0.379 0.454 0.471 1.000 1.000 0.471 0.455 0.309
0.171 0.255 0.263 0.392 0.390 0.263 0.255 0.239

Corn layer 1 image number 5
0.259 0.472 0.472 0.207 0.324 0.245 0.238 0.188
0.269 0.489 0.488 0.235 0.332 0.252 0.245 0.193
0.312 0.605 0.606 0.265 0.377 0.295 0.288 0.224
0.319 0.600 0.600 0.260 0.385 0.288 0.281 0.219
0.320 1.000 1.000 0.289 0.381 0.357 0.348 0.244
0.222 0.379 0.384 0.149 0.261 0.184 0.179 0.125
0.134 0.137 0.065 0.074 0.133 0.135 0.132 0.063
0.072 0.273 0.274 0.000 0.083 0.065 0.064 0.000

Corn layer 1 image number 6
0.357 0.396 0.423 0.470 0.594 0.516 0.501 0.332
0.408 0.425 0.440 0.709 0.506 0.440 0.428 0.301
0.456 0.479 0.496 1.000 0.592 0.496 0.482 0.330
0.272 0.388 0.400 0.585 0.451 0.400 0.390 0.241
0.225 0.325 0.333 0.364 0.395 0.358 0.350 0.172
0.125 0.197 0.203 0.290 0.236 0.203 0.197 0.070
0.357 0.463 0.479 1.000 0.596 0.479 0.466 0.303
0.171 0.252 0.260 0.394 0.292 0.260 0.254 0.236
```

```
Corn layer 1 image number 7
0.287 0.399 0.444 0.472 0.640 0.515 0.497 0.332
0.331 0.429 0.461 0.664 0.542 0.438 0.423 0.300
0.374 0.490 0.526 0.829 0.621 0.500 0.483 0.332
0.331 0.508 0.552 0.765 1.000 0.687 0.660 0.413
0.287 0.512 0.562 0.664 0.422 0.364 0.355 0.175
0.331 0.429 0.461 0.830 0.542 0.438 0.423 0.300
0.372 0.487 0.523 1.000 0.616 0.497 0.480 0.515
0.253 0.338 0.361 0.563 0.361 0.338 0.333 0.312

Corn layer 1 image number 8
0.065 0.061 0.070 0.072 0.064 0.063 0.062 0.147
0.000 0.000 0.000 0.250 0.250 0.000 0.000 0.074
0.000 0.000 0.000 0.256 0.256 0.000 0.000 0.063
0.234 0.247 0.253 1.000 1.000 0.253 0.247 0.070
0.234 0.247 0.253 1.000 1.000 0.253 0.247 0.071
0.000 0.000 0.000 0.273 0.273 0.000 0.000 0.071
0.000 0.000 0.000 0.250 0.250 0.000 0.000 0.068
0.000 0.000 0.000 0.222 0.222 0.000 0.000 0.061

Corn layer 1 image number 9
0.287 0.396 0.436 0.508 0.341 0.341 0.286 0.148
0.331 0.429 0.461 0.634 0.528 0.546 0.420 0.301
0.374 0.490 0.526 0.836 0.638 0.643 0.480 0.333
0.331 0.508 0.552 0.854 0.935 1.000 0.656 0.414
0.287 0.512 0.561 1.000 0.921 0.656 0.653 0.403
0.331 0.429 0.461 0.634 0.529 0.434 0.420 0.321
0.372 0.487 0.523 0.816 0.617 0.497 0.477 0.514
0.253 0.338 0.361 0.473 0.361 0.338 0.331 0.312

Corn layer 1 image number 10
0.176 0.189 0.209 0.528 0.073 0.073 0.073 0.148
0.202 0.205 0.219 0.649 0.239 0.222 0.222 0.302
0.238 0.242 0.253 1.000 0.278 0.250 0.245 0.331
0.129 0.147 0.156 0.544 0.168 0.159 0.157 0.242
0.070 0.072 0.081 0.397 0.084 0.074 0.074 0.171
0.000 0.000 0.000 0.266 0.000 0.000 0.000 0.068
0.235 0.239 0.250 1.000 0.274 0.240 0.235 0.070
0.000 0.000 0.000 0.272 0.000 0.000 0.000 0.070
```

Results A6.3 Maximum outputs in corner layer

The maximum results for the size 2 layer are shown in Results A6.4.

```
Size layer 1 image number 1
0.150 0.150 0.154 0.148 0.148 0.135
0.000 0.000 0.000 0.000 0.000 0.142
0.000 0.000 0.000 0.000 0.000 0.156
0.000 0.000 0.000 0.000 0.000 0.143
0.000 0.000 0.000 0.000 0.000 0.144
0.000 0.000 0.000 0.000 0.000 0.069

Size layer 1 image number 2
0.000 0.068 0.220 0.042 0.000 0.000
0.000 0.141 0.220 0.063 0.000 0.000
0.000 0.141 0.559 0.131 0.000 0.000
0.000 0.141 0.808 0.063 0.000 0.000
0.000 0.122 0.785 0.139 0.000 0.000
0.000 0.121 0.540 0.139 0.000 0.000
```

```
Size layer 1 image number 3
0.069 0.147 0.148 0.147 0.147 0.149
0.000 0.000 0.000 0.000 0.000 0.138
0.151 0.155 0.155 0.151 0.155 0.143
0.546 0.570 1.000 0.837 0.553 0.549
0.145 0.146 0.149 0.069 0.143 0.069
0.000 0.000 0.000 0.000 0.000 0.000

Size layer 1 image number 4
0.143 0.147 0.147 0.147 0.147 0.150
0.000 0.000 0.000 0.000 0.000 0.072
0.150 0.153 0.156 0.152 0.152 0.071
0.544 0.546 1.000 1.000 0.546 0.544
0.143 0.146 0.146 0.147 0.146 0.069
0.000 0.000 0.000 0.000 0.000 0.043

Size layer 1 image number 5
0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.143 0.000 0.000
0.000 0.000 0.149 0.143 0.140 0.000
0.000 0.000 0.132 0.634 0.137 0.000
0.144 0.150 0.150 0.559 0.133 0.045
0.522 0.501 0.451 0.214 0.485 0.483

Size layer 1 image number 6
0.151 0.153 0.154 0.151 0.151 0.069
0.000 0.000 0.000 0.000 0.000 0.000
0.152 0.153 0.153 0.152 0.152 0.069
0.657 0.677 1.000 0.255 0.246 0.246
0.143 0.147 0.142 0.147 0.147 0.068
0.000 0.000 0.000 0.000 0.000 0.027

Size layer 1 image number 7
0.153 0.153 0.154 0.151 0.151 0.069
0.000 0.000 0.000 0.000 0.000 0.000
0.076 0.157 0.155 0.150 0.147 0.139
0.655 0.677 1.000 0.632 0.622 0.630
0.146 0.148 0.155 0.147 0.147 0.156
0.000 0.000 0.000 0.000 0.000 0.069

Size layer 1 image number 8
0.127 0.127 0.127 0.063 0.130 0.063
0.000 0.000 0.000 0.000 0.000 0.127
0.000 0.000 0.000 0.000 0.000 0.061
0.000 0.000 0.000 0.000 0.000 0.114
0.000 0.000 0.000 0.000 0.000 0.114
0.000 0.000 0.000 0.000 0.000 0.114

Size layer 1 image number 9
0.149 0.149 0.152 0.148 0.148 0.135
0.000 0.000 0.000 0.000 0.000 0.143
0.048 0.072 0.149 0.145 0.149 0.146
0.655 0.662 1.000 0.636 0.628 0.622
0.149 0.149 0.155 0.148 0.148 0.155
0.000 0.000 0.000 0.000 0.000 0.069

Size layer 1 image number 10
0.142 0.142 0.044 0.068 0.141 0.137
0.000 0.000 0.000 0.000 0.000 0.042
0.145 0.145 0.045 0.071 0.147 0.071
0.501 0.509 1.000 0.513 0.503 0.501
0.127 0.127 0.040 0.064 0.133 0.065
0.000 0.000 0.000 0.000 0.000 0.040
```

Results A6.4 Maximum outputs in size 2 layer

Before the system can produce a structural decomposition of the images, the feature neurons must be assigned to features in the image set. One way of determining which features have been learnt is to examine the synapse files. An example of part of a synapse file is shown in Results A6.5.

```
Layer 1
Feature 1
g  2 d  1 s 0.04496907
g  2 d  2 s 0.21728389
g  2 d  3 s 0.04497505
g  2 d  4 s 0.21728334
g  2 d  5 s 0.47317559
g  2 d  6 s 0.21728331
g  2 d  7 s 0.47316962
g  2 d  8 s 0.16468577
g  2 d  9 s 0.02815758
g  2 d 10 s 0.21400869
g  2 d 11 s 0.99998701
g  2 d 12 s 0.21399400
g  2 d 13 s 0.12164043
g  2 d 14 s 0.12164816
g  2 d 15 s 0.66929042
g  2 d 16 s 0.66928220
g  2 d 17 s 0.34213549
g  1 d  1 s 0.14856979
g  1 d  2 s 0.14841679
g  1 d  3 s 0.14840955
g  1 d  4 s 0.14841571
g  1 d  5 s 0.01125680
g  1 d  6 s 0.14841551
g  1 d  7 s 0.14844272
g  1 d  8 s 0.11418228
g  1 d  9 s 0.57671309
g  1 d 10 s 0.08093292
g  1 d 11 s 0.08094688
g  1 d 12 s 0.57677269
g  1 d 13 s 0.99981499
g  1 d 14 s 0.27916259
g  1 d 15 s 0.27905202
g  1 d 16 s 0.27920380
g  1 d 17 s 0.75210404
```

Results A6.5 Part of a size layer synapse file

By examining these values, it is easy to determine that the feature identifies a strong output in both feature numbers 11 and 13 in the primitive layer. Using the primitive layer features shown in Figure A6.2, it is clear that the classified feature in the size layer consists of a tee and a corner. This analysis can be continued for all the features in the size layer giving the features shown in Figure A6.3.

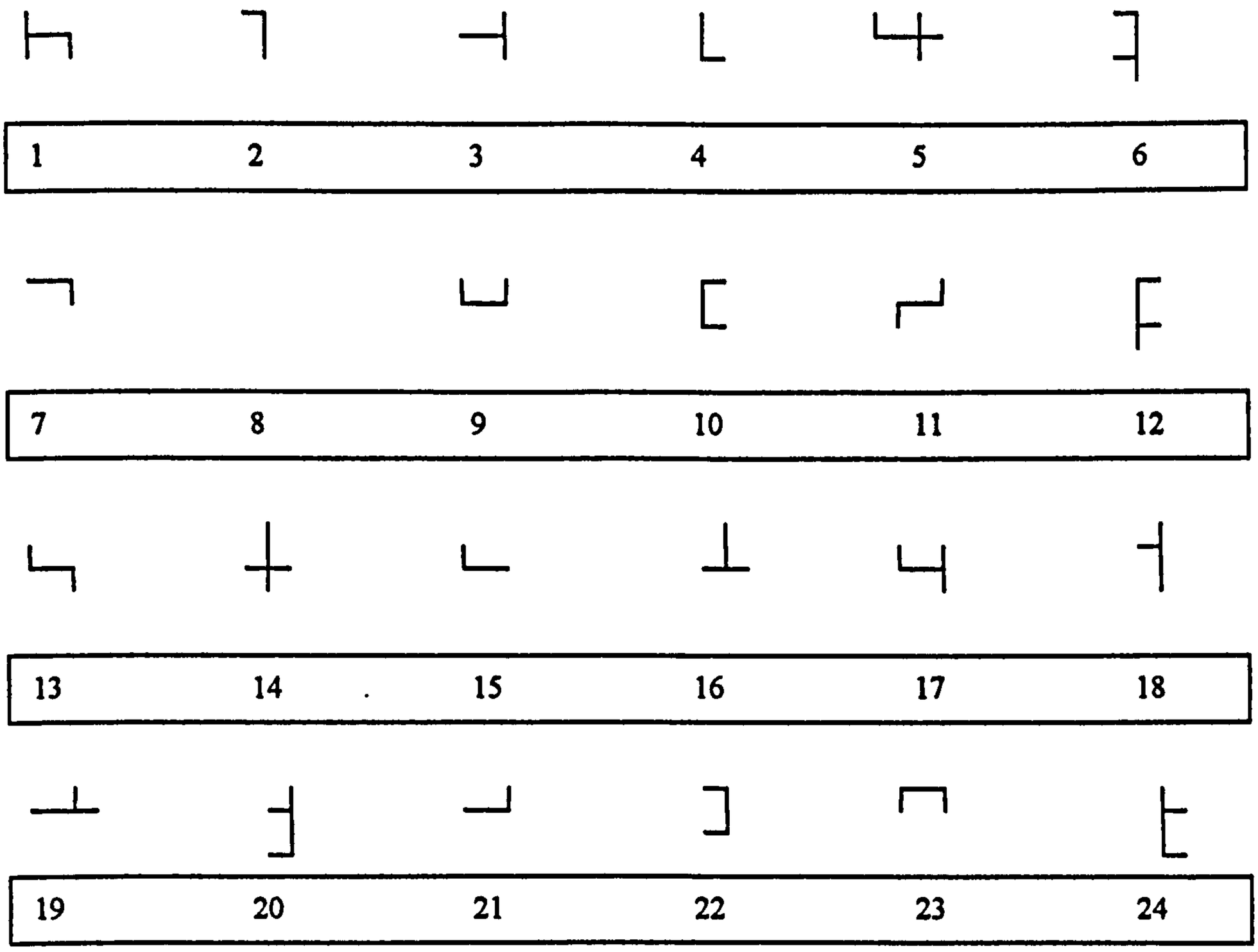


Figure A6.3 Learnt size layer features

A clearly identifiable feature was not learnt by feature neuron 8.

In a similar manner the features learnt by the corner layer are identified and are shown in Figure A6.4.

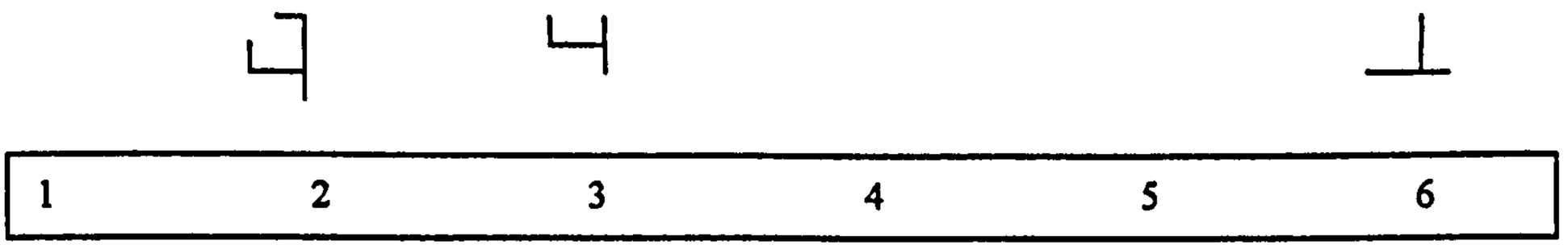


Figure A6.4 Learnt corner layer features

Features 1, 9 and 12 did not learn fully classified size layer features. However, they were still able to give a strong response to the size layer. When constructing the structures file, feature 1 was identified as number 0, and the features 9 and 12 were identified as corners.

The structure file used for the structural decomposition is shown in Results A6.6.

```
* Structure file for testing
PRIMITIVE 1
17
1       "Left horizontal line"
2       "Left 45 degree line"
3       "Bottom vertical line"
4       "Right 135 degree line"
5       "Right horizontal line"
6       "Right 225 degree line"
7       "Top vertical line"
8       "Left 315 degree line"
9       "Bottom left corner"
10      "Bottom right corner"
11      "Top right corner"
12      "Top left corner"
13      "Left tee"
14      "Bottom tee"
15      "Right tee"
16      "Top tee"
17      "Cross"
```

```
SIZE 1
24
1       "right of tee and left of down corner"
2       "Right: above edge and below corner"
3       "right of edge and left of tee"
4       "Left: below tee and above corner"
5       "right of corner and left of tee"
6       "Right: below corner and above tee"
7       "Top: right of edge and left of corner"
8       "No feature"
9       "Bottom: right of corner and left of corner"
10      "Left: below corner and above corner"
11      "left of bottom corner and right of top corner"
12      "Left: below corner and above tee"
13      "right of bottom corner and left of top corner"
14      "below edge and above cross"
15      "Bottom: right of corner and left of edge"
16      "below edge and above tee"
17      "right of bottom corner and left of tee"
18      "Right: below tee and above edge"
19      "Bottom: right of edge and left of tee"
20      "Right: below tee and above corner"
21      "Bottom: right of edge and left of corner"
22      "Right: below corner and above corner"
23      "Top: right of corner and left of corner"
24      "Left: below tee and above corner"
CORN 1
16
1       "low input number 0 <<<<<<<<<<"
2       "closed top left tee"
3       "open bottom left tee"
4       "No feature"
5       "No feature"
6       "open top left tee"
7       "number 7 <<<<<<<<<<"
8       "open bottom right tee"
9       "low input bottom left corner"
10      "open top left tee"
11      "top right corner"
12      "low input top left corner"
13      "bottom left corner"
14      "open top left cross"
15      "open bottom left tee"
16      "open top left corner"
SIZE2 1
12
1       "No feature"
2       "number 5 <<<<<<<<<<"
3       "No feature"
4       "number 6 <<<<<<<<<<"
5       "number 4 <<<<<<<<<<"
6       "number 3 <<<<<<<<<<"
7       "No feature"
8       "number 2 <<<<<<<<<<"
9       "number 1 <<<<<<<<<<"
10      "number 8 <<<<<<<<<<"
11      "No feature"
12      "number 9 <<<<<<<<<<"
```

Results A6.6 Structure file used

The structural decomposition of the images is shown in Results A6.7.

```
MAX OUTPUT Primitive layer 1 image number 1
feature number 9 row 2 column 2 value 1.000000 Bottom left corner

MAX OUTPUT Size layer 1 image number 1
feature number 9 row 1 column 5 value 1.000000 Bottom: right of corner
and left of corner

MAX OUTPUT Corn layer 1 image number 1
feature number 1 row 5 column 5 value 1.000000 low input number 0
<<<<<<<<<<
MAX OUTPUT Size2 layer 1 image number 1
feature number 2 row 4 column 6 value 0.155667 number 5 <<<<<<<<<<

FULL OUTPUT Primitive layer 1 image number 1
feature number 9 row 2 column 2 Bottom left corner

FULL OUTPUT Primitive layer 1 image number 1
feature number 10 row 2 column 10 Bottom right corner

FULL OUTPUT Primitive layer 1 image number 1
feature number 11 row 10 column 10 Top right corner

FULL OUTPUT Primitive layer 1 image number 1
feature number 12 row 10 column 2 Top left corner

FULL OUTPUT Size layer 1 image number 1
feature number 9 row 1 column 5 Bottom: right of corner and left of
corner

FULL OUTPUT Size layer 1 image number 1
feature number 10 row 5 column 1 Left: below corner and above corner

FULL OUTPUT Size layer 1 image number 1
feature number 22 row 5 column 9 Right: below corner and above corner

FULL OUTPUT Size layer 1 image number 1
feature number 23 row 9 column 5 Top: right of corner and left of corner

FULL OUTPUT Corn layer 1 image number 1
feature number 1 row 5 column 5 low input number 0 <<<<<<<<<<

FULL OUTPUT Size2 layer 1 image number 1
no feature found

MAX OUTPUT Primitive layer 1 image number 2
feature number 1 row 2 column 2 value 1.000000 Left horizontal line

MAX OUTPUT Size layer 1 image number 2
feature number 16 row 4 column 5 value 1.000000 below edge and above tee

MAX OUTPUT Corn layer 1 image number 2
feature number 6 row 3 column 3 value 1.000000 open top left tee

MAX OUTPUT Size2 layer 1 image number 2
feature number 9 row 3 column 3 value 0.807680 number 1 <<<<<<<<<<

FULL OUTPUT Primitive layer 1 image number 2
feature number 1 row 2 column 2 Left horizontal line

FULL OUTPUT Primitive layer 1 image number 2
feature number 1 row 2 column 3 Left horizontal line
```

```
FULL OUTPUT Primitive layer 1 image number 2
feature number 5 row 2 column 9 Right horizontal line

FULL OUTPUT Primitive layer 1 image number 2
feature number 5 row 2 column 10 Right horizontal line

FULL OUTPUT Primitive layer 1 image number 2
feature number 7 row 9 column 6 Top vertical line

FULL OUTPUT Primitive layer 1 image number 2
feature number 7 row 10 column 6 Top vertical line

FULL OUTPUT Primitive layer 1 image number 2
feature number 7 row 11 column 6 Top vertical line

FULL OUTPUT Primitive layer 1 image number 2
feature number 14 row 2 column 6 Bottom tee

FULL OUTPUT Size layer 1 image number 2
feature number 16 row 4 column 5 below edge and above tee

FULL OUTPUT Size layer 1 image number 2
feature number 16 row 5 column 5 below edge and above tee

FULL OUTPUT Size layer 1 image number 2
feature number 19 row 1 column 3 Bottom: right of edge and left of tee

FULL OUTPUT Size layer 1 image number 2
feature number 19 row 1 column 4 Bottom: right of edge and left of tee

FULL OUTPUT Corn layer 1 image number 2
feature number 6 row 3 column 3 open top left tee

FULL OUTPUT Corn layer 1 image number 2
feature number 6 row 4 column 3 open top left tee

FULL OUTPUT Size2 layer 1 image number 2
no feature found

MAX OUTPUT Primitive layer 1 image number 3
feature number 1 row 10 column 1 value 1.000000 Left horizontal line

MAX OUTPUT Size layer 1 image number 3
feature number 7 row 9 column 5 value 1.000000 Top: right of edge and
left of corner

MAX OUTPUT Corn layer 1 image number 3
feature number 7 row 6 column 4 value 1.000000 number 7 <<<<<<<<<<

MAX OUTPUT Size2 layer 1 image number 3
feature number 8 row 3 column 3 value 1.000000 number 2 <<<<<<<<<<

FULL OUTPUT Primitive layer 1 image number 3
feature number 1 row 10 column 1 Left horizontal line

FULL OUTPUT Primitive layer 1 image number 3
feature number 1 row 10 column 2 Left horizontal line

FULL OUTPUT Primitive layer 1 image number 3
feature number 1 row 10 column 3 Left horizontal line

FULL OUTPUT Primitive layer 1 image number 3
feature number 5 row 2 column 9 Right horizontal line

FULL OUTPUT Primitive layer 1 image number 3
feature number 5 row 2 column 10 Right horizontal line

FULL OUTPUT Primitive layer 1 image number 3
feature number 5 row 2 column 11 Right horizontal line
```

```
FULL OUTPUT Primitive layer 1 image number 3
feature number 9 row 2 column 2 Bottom left corner

FULL OUTPUT Primitive layer 1 image number 3
feature number 10 row 6 column 10 Bottom right corner

FULL OUTPUT Primitive layer 1 image number 3
feature number 11 row 10 column 10 Top right corner

FULL OUTPUT Primitive layer 1 image number 3
feature number 12 row 6 column 2 Top left corner

FULL OUTPUT Size layer 1 image number 3
feature number 7 row 9 column 5 Top: right of edge and left of corner

FULL OUTPUT Size layer 1 image number 3
feature number 7 row 9 column 6 Top: right of edge and left of corner

FULL OUTPUT Size layer 1 image number 3
feature number 10 row 3 column 1 Left: below corner and above corner

FULL OUTPUT Size layer 1 image number 3
feature number 11 row 5 column 5 left of bottom corner and right of top
corner

FULL OUTPUT Size layer 1 image number 3
feature number 15 row 1 column 4 Bottom: right of corner and left of edge

FULL OUTPUT Size layer 1 image number 3
feature number 15 row 1 column 5 Bottom: right of corner and left of edge

FULL OUTPUT Size layer 1 image number 3
feature number 22 row 7 column 9 Right: below corner and above corner

FULL OUTPUT Corn layer 1 image number 3
feature number 7 row 6 column 4 number 7 <<<<<<<<<<

FULL OUTPUT Corn layer 1 image number 3
feature number 7 row 6 column 5 number 7 <<<<<<<<<<

FULL OUTPUT Corn layer 1 image number 3
feature number 12 row 4 column 4 low input top left corner

FULL OUTPUT Corn layer 1 image number 3
feature number 16 row 2 column 4 open top left corner

FULL OUTPUT Size2 layer 1 image number 3
feature number 8 row 3 column 3 number 2 <<<<<<<<<<

MAX OUTPUT Primitive layer 1 image number 4
feature number 1 row 2 column 1 value 1.000000 Left horizontal line

MAX OUTPUT Size layer 1 image number 4
feature number 3 row 5 column 5 value 1.000000 right of edge and left of
tee

MAX OUTPUT Corn layer 1 image number 4
feature number 10 row 6 column 4 value 1.000000 open top left tee

MAX OUTPUT Size2 layer 1 image number 4
feature number 6 row 3 column 3 value 1.000000 number 3 <<<<<<<<<<

FULL OUTPUT Primitive layer 1 image number 4
feature number 1 row 2 column 1 Left horizontal line

FULL OUTPUT Primitive layer 1 image number 4
feature number 1 row 2 column 2 Left horizontal line

FULL OUTPUT Primitive layer 1 image number 4
feature number 1 row 2 column 3 Left horizontal line
```

```
FULL OUTPUT Primitive layer 1 image number 4
feature number 1 row 6 column 1 Left horizontal line

FULL OUTPUT Primitive layer 1 image number 4
feature number 1 row 6 column 2 Left horizontal line

FULL OUTPUT Primitive layer 1 image number 4
feature number 1 row 6 column 3 Left horizontal line

FULL OUTPUT Primitive layer 1 image number 4
feature number 1 row 10 column 1 Left horizontal line

FULL OUTPUT Primitive layer 1 image number 4
feature number 1 row 10 column 2 Left horizontal line

FULL OUTPUT Primitive layer 1 image number 4
feature number 1 row 10 column 3 Left horizontal line

FULL OUTPUT Primitive layer 1 image number 4
feature number 10 row 2 column 10 Bottom right corner

FULL OUTPUT Primitive layer 1 image number 4
feature number 11 row 10 column 10 Top right corner

FULL OUTPUT Primitive layer 1 image number 4
feature number 15 row 6 column 10 Right tee

FULL OUTPUT Size layer 1 image number 4
feature number 3 row 5 column 5 right of edge and left of tee

FULL OUTPUT Size layer 1 image number 4
feature number 3 row 5 column 6 right of edge and left of tee

FULL OUTPUT Size layer 1 image number 4
feature number 6 row 7 column 9 Right: below corner and above tee

FULL OUTPUT Size layer 1 image number 4
feature number 7 row 9 column 5 Top: right of edge and left of corner

FULL OUTPUT Size layer 1 image number 4
feature number 7 row 9 column 6 Top: right of edge and left of corner

FULL OUTPUT Size layer 1 image number 4
feature number 20 row 3 column 9 Right: below tee and above corner

FULL OUTPUT Size layer 1 image number 4
feature number 21 row 1 column 5 Bottom: right of edge and left of corner

FULL OUTPUT Size layer 1 image number 4
feature number 21 row 1 column 6 Bottom: right of edge and left of corner

FULL OUTPUT Corn layer 1 image number 4
feature number 10 row 6 column 4 open top left tee

FULL OUTPUT Corn layer 1 image number 4
feature number 10 row 6 column 5 open top left tee

FULL OUTPUT Corn layer 1 image number 4
feature number 15 row 2 column 4 open bottom left tee

FULL OUTPUT Corn layer 1 image number 4
feature number 15 row 2 column 5 open bottom left tee

FULL OUTPUT Size2 layer 1 image number 4
feature number 6 row 3 column 3 number 3 <<<<<<<<<<

FULL OUTPUT Size2 layer 1 image number 4
feature number 6 row 3 column 4 number 3 <<<<<<<<<<
```

```
MAX OUTPUT Primitive layer 1 image number 5
feature number 5 row 4 column 11 value 1.000000 Right horizontal line

MAX OUTPUT Size layer 1 image number 5
feature number 4 row 6 column 1 value 1.000000 Left: below tee and above
corner

MAX OUTPUT Corn layer 1 image number 5
feature number 14 row 4 column 2 value 1.000000 open top left cross

MAX OUTPUT Size2 layer 1 image number 5
feature number 5 row 3 column 4 value 0.633802 number 4 <<<<<<<<<<

FULL OUTPUT Primitive layer 1 image number 5
feature number 5 row 4 column 11 Right horizontal line

FULL OUTPUT Primitive layer 1 image number 5
feature number 7 row 8 column 7 Top vertical line

FULL OUTPUT Primitive layer 1 image number 5
feature number 7 row 11 column 2 Top vertical line

FULL OUTPUT Primitive layer 1 image number 5
feature number 7 row 12 column 2 Top vertical line

FULL OUTPUT Primitive layer 1 image number 5
feature number 9 row 4 column 2 Bottom left corner

FULL OUTPUT Primitive layer 1 image number 5
feature number 17 row 4 column 7 Cross

FULL OUTPUT Size layer 1 image number 5
feature number 4 row 6 column 1 Left: below tee and above corner

FULL OUTPUT Size layer 1 image number 5
feature number 4 row 7 column 1 Left: below tee and above corner

FULL OUTPUT Size layer 1 image number 5
feature number 5 row 3 column 3 right of corner and left of tee

FULL OUTPUT Size layer 1 image number 5
feature number 5 row 3 column 4 right of corner and left of tee

FULL OUTPUT Size layer 1 image number 5
feature number 14 row 5 column 6 below edge and above cross

FULL OUTPUT Corn layer 1 image number 5
feature number 14 row 4 column 2 open top left cross

FULL OUTPUT Corn layer 1 image number 5
feature number 14 row 4 column 3 open top left cross

FULL OUTPUT Size2 layer 1 image number 5
no feature found

MAX OUTPUT Primitive layer 1 image number 6
feature number 1 row 2 column 1 value 1.000000 Left horizontal line

MAX OUTPUT Size layer 1 image number 6
feature number 10 row 7 column 1 value 1.000000 Left: below corner and
above corner

MAX OUTPUT Corn layer 1 image number 6
feature number 11 row 2 column 4 value 1.000000 top right corner

MAX OUTPUT Size2 layer 1 image number 6
feature number 2 row 3 column 3 value 1.000000 number 5 <<<<<<<<<<

FULL OUTPUT Primitive layer 1 image number 6
feature number 1 row 2 column 1 Left horizontal line
```

```
FULL OUTPUT Primitive layer 1 image number 6
feature number 1 row 2 column 2 Left horizontal line

FULL OUTPUT Primitive layer 1 image number 6
feature number 1 row 2 column 3 Left horizontal line

FULL OUTPUT Primitive layer 1 image number 6
feature number 5 row 10 column 9 Right horizontal line

FULL OUTPUT Primitive layer 1 image number 6
feature number 5 row 10 column 10 Right horizontal line

FULL OUTPUT Primitive layer 1 image number 6
feature number 5 row 10 column 11 Right horizontal line

FULL OUTPUT Primitive layer 1 image number 6
feature number 9 row 6 column 2 Bottom left corner

FULL OUTPUT Primitive layer 1 image number 6
feature number 10 row 2 column 10 Bottom right corner

FULL OUTPUT Primitive layer 1 image number 6
feature number 11 row 6 column 10 Top right corner

FULL OUTPUT Primitive layer 1 image number 6
feature number 12 row 10 column 2 Top left corner

FULL OUTPUT Size layer 1 image number 6
feature number 10 row 7 column 1 Left: below corner and above corner

FULL OUTPUT Size layer 1 image number 6
feature number 13 row 5 column 5 right of bottom corner and left of top
corner

FULL OUTPUT Size layer 1 image number 6
feature number 21 row 1 column 5 Bottom: right of edge and left of corner

FULL OUTPUT Size layer 1 image number 6
feature number 21 row 1 column 6 Bottom: right of edge and left of corner

FULL OUTPUT Size layer 1 image number 6
feature number 22 row 3 column 9 Right: below corner and above corner

FULL OUTPUT Corn layer 1 image number 6
feature number 11 row 2 column 4 top right corner

FULL OUTPUT Corn layer 1 image number 6
feature number 13 row 6 column 4 bottom left corner

FULL OUTPUT Size2 layer 1 image number 6
feature number 2 row 3 column 3 number 5 <<<<<<<<<<

MAX OUTPUT Primitive layer 1 image number 7
feature number 5 row 10 column 9 value 1.000000 Right horizontal line

MAX OUTPUT Size layer 1 image number 7
feature number 1 row 5 column 5 value 1.000000 right of tee and left of
down corner

MAX OUTPUT Corn layer 1 image number 7
feature number 1 row 5 column 5 value 1.000000 low input number 0
<<<<<<<<<<

MAX OUTPUT Size2 layer 1 image number 7
feature number 4 row 3 column 3 value 1.000000 number 6 <<<<<<<<<<

FULL OUTPUT Primitive layer 1 image number 7
feature number 5 row 10 column 9 Right horizontal line
```

```
FULL OUTPUT Primitive layer 1 image number 7
feature number 5 row 10 column 10 Right horizontal line

FULL OUTPUT Primitive layer 1 image number 7
feature number 5 row 10 column 11 Right horizontal line

FULL OUTPUT Primitive layer 1 image number 7
feature number 9 row 2 column 2 Bottom left corner

FULL OUTPUT Primitive layer 1 image number 7
feature number 10 row 2 column 10 Bottom right corner

FULL OUTPUT Primitive layer 1 image number 7
feature number 11 row 6 column 10 Top right corner

FULL OUTPUT Primitive layer 1 image number 7
feature number 12 row 10 column 2 Top left corner

FULL OUTPUT Primitive layer 1 image number 7
feature number 13 row 6 column 2 Left tee

FULL OUTPUT Size layer 1 image number 7
feature number 1 row 5 column 5 right of tee and left of down corner

FULL OUTPUT Size layer 1 image number 7
feature number 9 row 1 column 5 Bottom: right of corner and left of
corner

FULL OUTPUT Size layer 1 image number 7
feature number 12 row 7 column 1 Left: below corner and above tee

FULL OUTPUT Size layer 1 image number 7
feature number 22 row 3 column 9 Right: below corner and above corner

FULL OUTPUT Size layer 1 image number 7
feature number 24 row 3 column 1 Left: below tee and above corner

FULL OUTPUT Corn layer 1 image number 7
feature number 1 row 5 column 5 low input number 0 <<<<<<<<<<

FULL OUTPUT Corn layer 1 image number 7
feature number 8 row 2 column 4 open bottom right tee

FULL OUTPUT Size2 layer 1 image number 7
feature number 4 row 3 column 3 number 6 <<<<<<<<<<

MAX OUTPUT Primitive layer 1 image number 8
feature number 1 row 10 column 1 value 1.000000 Left horizontal line

MAX OUTPUT Size layer 1 image number 8
feature number 2 row 5 column 9 value 1.000000 Right: above edge and
below corner

MAX OUTPUT Corn layer 1 image number 8
feature number 7 row 4 column 4 value 1.000000 number 7 <<<<<<<<<<

MAX OUTPUT Size2 layer 1 image number 8
feature number 9 row 6 column 5 value 0.130019 number 1 <<<<<<<<<<

FULL OUTPUT Primitive layer 1 image number 8
feature number 1 row 10 column 1 Left horizontal line

FULL OUTPUT Primitive layer 1 image number 8
feature number 1 row 10 column 2 Left horizontal line

FULL OUTPUT Primitive layer 1 image number 8
feature number 1 row 10 column 3 Left horizontal line

FULL OUTPUT Primitive layer 1 image number 8
feature number 3 row 1 column 10 Bottom vertical line
```

```
FULL OUTPUT Primitive layer 1 image number 8
feature number 3 row 2 column 10 Bottom vertical line

FULL OUTPUT Primitive layer 1 image number 8
feature number 3 row 3 column 10 Bottom vertical line

FULL OUTPUT Primitive layer 1 image number 8
feature number 11 row 10 column 10 Top right corner

FULL OUTPUT Size layer 1 image number 8
feature number 2 row 5 column 9 Right: above edge and below corner

FULL OUTPUT Size layer 1 image number 8
feature number 2 row 6 column 9 Right: above edge and below corner

FULL OUTPUT Size layer 1 image number 8
feature number 7 row 9 column 5 Top: right of edge and left of corner

FULL OUTPUT Size layer 1 image number 8
feature number 7 row 9 column 6 Top: right of edge and left of corner

FULL OUTPUT Corn layer 1 image number 8
feature number 7 row 4 column 4 number 7 <<<<<<<<<<

FULL OUTPUT Corn layer 1 image number 8
feature number 7 row 4 column 5 number 7 <<<<<<<<<<

FULL OUTPUT Corn layer 1 image number 8
feature number 7 row 5 column 4 number 7 <<<<<<<<<<

FULL OUTPUT Corn layer 1 image number 8
feature number 7 row 5 column 5 number 7 <<<<<<<<<<

FULL OUTPUT Size2 layer 1 image number 8
no feature found

MAX OUTPUT Primitive layer 1 image number 9
feature number 9 row 2 column 2 value 1.000000 Bottom left corner

MAX OUTPUT Size layer 1 image number 9
feature number 6 row 7 column 9 value 1.000000 Right: below corner and
above tee

MAX OUTPUT Corn layer 1 image number 9
feature number 5 row 5 column 6 value 1.000000 No feature

MAX OUTPUT Size2 layer 1 image number 9
feature number 10 row 3 column 3 value 1.000000 number 8 <<<<<<<<<<

FULL OUTPUT Primitive layer 1 image number 9
feature number 9 row 2 column 2 Bottom left corner

FULL OUTPUT Primitive layer 1 image number 9
feature number 10 row 2 column 10 Bottom right corner

FULL OUTPUT Primitive layer 1 image number 9
feature number 11 row 10 column 10 Top right corner

FULL OUTPUT Primitive layer 1 image number 9
feature number 12 row 10 column 2 Top left corner

FULL OUTPUT Primitive layer 1 image number 9
feature number 13 row 6 column 2 Left tee

FULL OUTPUT Primitive layer 1 image number 9
feature number 15 row 6 column 10 Right tee

FULL OUTPUT Size layer 1 image number 9
feature number 6 row 7 column 9 Right: below corner and above tee
```

```
FULL OUTPUT Size layer 1 image number 9
feature number 9 row 1 column 5 Bottom: right of corner and left of
corner

FULL OUTPUT Size layer 1 image number 9
feature number 12 row 7 column 1 Left: below corner and above tee

FULL OUTPUT Size layer 1 image number 9
feature number 20 row 3 column 9 Right: below tee and above corner

FULL OUTPUT Size layer 1 image number 9
feature number 23 row 9 column 5 Top: right of corner and left of corner

FULL OUTPUT Size layer 1 image number 9
feature number 24 row 3 column 1 Left: below tee and above corner

FULL OUTPUT Corn layer 1 image number 9
feature number 5 row 5 column 6 No feature

FULL OUTPUT Corn layer 1 image number 9
feature number 12 row 4 column 4 low input top left corner

FULL OUTPUT Size2 layer 1 image number 9
feature number 10 row.3 column 3 number 8 <<<<<<<<<<

MAX OUTPUT Primitive layer 1 image number 10
feature number 3 row 1 column 10 value 1.000000 Bottom vertical line

MAX OUTPUT Size layer 1 image number 10
feature number 6 row 7 column 9 value 1.000000 Right: below corner and
above tee

MAX OUTPUT Corn layer 1 image number 10
feature number 2 row 6 column 4 value 1.000000 closed top left tee

MAX OUTPUT Size2 layer 1 image number 10
feature number 12 row 3 column 3 value 1.000000 number 9 <<<<<<<<<<

FULL OUTPUT Primitive layer 1 image number 10
feature number 3 row 1 column 10 Bottom vertical line

FULL OUTPUT Primitive layer 1 image number 10
feature number 3 row 2 column 10 Bottom vertical line

FULL OUTPUT Primitive layer 1 image number 10
feature number 9 row 6 column 2 Bottom left corner

FULL OUTPUT Primitive layer 1 image number 10
feature number 11 row 10 column 10 Top right corner

FULL OUTPUT Primitive layer 1 image number 10
feature number 12 row 10 column 2 Top left corner

FULL OUTPUT Primitive layer 1 image number 10
feature number 15 row 6 column 10 Right tee

FULL OUTPUT Size layer 1 image number 10
feature number 6 row 7 column 9 Right: below corner and above tee

FULL OUTPUT Size layer 1 image number 10
feature number 10 row 7 column 1 Left: below corner and above corner

FULL OUTPUT Size layer 1 image number 10
feature number 17 row 5 column 5 right of bottom corner and left of tee

FULL OUTPUT Size layer 1 image number 10
feature number 18 row 3 column 9 Right: below tee and above edge

FULL OUTPUT Size layer 1 image number 10
feature number 23 row 9 column 5 Top: right of corner and left of corner
```

```
FULL OUTPUT Corn layer 1 image number 10
feature number 2 row 6 column 4 closed top left tee

FULL OUTPUT Corn layer 1 image number 10
feature number 3 row 2 column 4 open bottom left tee

FULL OUTPUT Size2 layer 1 image number 10
feature number 12 row 3 column 3 number 9 <<<<<<<<<<
```

Results A6.7 Structural decomposition

The classification experiment was repeated using an image set that was two pixels wider and taller than the training set. A retina of size 20 × 20 was used for these tests. The maximum output values for the primitive layer are shown in Results A6.8.

```
Layer 1 image number 1
0.319 0.473 0.217 0.217 0.217 0.217 0.217 0.217 0.217 0.217 0.217 0.473
0.319 0.319
0.237 0.722 0.165 0.217 0.217 0.217 0.217 0.217 0.217 0.217 0.165 0.722
0.237 0.319
0.722 1.000 0.722 0.722 0.493 0.669 0.669 0.669 0.493 0.722 0.722 1.000
0.722 0.473
0.165 0.722 0.112 0.148 0.148 0.217 0.217 0.217 0.148 0.148 0.112 0.722
0.148 0.237
0.217 0.722 0.148 0.148 0.148 0.217 0.217 0.217 0.165 0.148 0.148 0.722
0.217 0.217
0.217 0.493 0.148 0.148 0.148 0.217 0.217 0.217 0.165 0.148 0.148 0.493
0.217 0.217
0.217 0.669 0.217 0.217 0.217 0.342 0.342 0.342 0.237 0.217 0.217 0.669
0.217 0.217
0.217 0.669 0.217 0.217 0.217 0.342 0.342 0.342 0.237 0.217 0.217 0.669
0.217 0.217
0.217 0.669 0.217 0.217 0.217 0.342 0.342 0.342 0.237 0.217 0.217 0.669
0.217 0.217
0.217 0.493 0.148 0.165 0.165 0.237 0.237 0.237 0.148 0.148 0.148 0.493
0.217 0.217
0.217 0.722 0.148 0.148 0.148 0.217 0.217 0.217 0.148 0.148 0.148 0.722
0.217 0.217
0.165 0.722 0.112 0.148 0.148 0.217 0.217 0.217 0.148 0.148 0.114 0.722
0.148 0.237
0.722 1.000 0.722 0.722 0.493 0.669 0.669 0.669 0.493 0.722 0.722 1.000
0.722 0.473
0.237 0.722 0.148 0.217 0.217 0.217 0.217 0.217 0.217 0.217 0.148 0.722
0.237 0.319
```

Layer 1 image number 2
0.342 0.342 0.342 0.342 0.342 0.319 0.722 0.342 0.319 0.319 0.342 0.342
0.342 0.342
0.342 0.342 0.342 0.342 0.319 0.342 1.000 0.319 0.319 0.319 0.342 0.342
0.342 0.342
0.342 0.342 0.342 0.342 0.319 0.319 1.000 0.319 0.319 0.319 0.342 0.342
0.342 0.342
0.342 0.342 0.342 0.342 0.237 0.217 1.000 0.217 0.237 0.319 0.342 0.342
0.342 0.342
0.342 0.342 0.342 0.342 0.217 0.217 0.722 0.217 0.217 0.319 0.342 0.342
0.342 0.342
0.342 0.342 0.342 0.237 0.217 0.217 0.669 0.217 0.217 0.217 0.342 0.342
0.342 0.342
0.342 0.342 0.342 0.237 0.217 0.217 0.669 0.217 0.217 0.217 0.342 0.342
0.342 0.342
0.342 0.342 0.342 0.237 0.217 0.217 0.669 0.217 0.217 0.217 0.342 0.342
0.342 0.342
0.342 0.342 0.342 0.237 0.217 0.217 0.669 0.217 0.217 0.217 0.342 0.342
0.342 0.342
0.342 0.342 0.342 0.217 0.237 0.148 0.473 0.148 0.217 0.217 0.319 0.319
0.319 0.473
0.342 0.319 0.319 0.217 0.148 0.148 0.722 0.148 0.148 0.217 0.319 0.319
0.319 0.342
0.319 0.342 0.319 0.148 0.148 0.101 0.473 0.101 0.148 0.165 0.319 0.319
0.342 0.342
0.722 1.000 1.000 0.722 0.473 0.493 1.000 0.493 0.473 0.722 1.000 1.000
0.722 0.473
0.342 0.319 0.319 0.217 0.217 0.148 0.473 0.148 0.217 0.217 0.319 0.319
0.342 0.342

Layer 1 image number 3
0.319 0.319 0.217 0.217 0.217 0.217 0.217 0.217 0.217 0.217 0.217 0.473
0.319 0.319
0.342 0.319 0.217 0.217 0.217 0.217 0.217 0.217 0.217 0.217 0.165 0.722
0.237 0.319
1.000 1.000 1.000 0.722 0.669 0.669 0.669 0.669 0.493 0.722 0.722 1.000
0.722 0.473
0.319 0.319 0.217 0.217 0.217 0.217 0.217 0.217 0.148 0.148 0.112 0.722
0.148 0.237
0.319 0.237 0.237 0.165 0.148 0.148 0.148 0.148 0.101 0.101 0.101 0.473
0.217 0.217
0.319 0.319 0.217 0.148 0.148 0.148 0.148 0.148 0.101 0.101 0.101 0.473
0.217 0.217
0.237 0.722 0.165 0.217 0.217 0.217 0.217 0.217 0.148 0.148 0.114 0.722
0.148 0.237
0.722 1.000 0.722 0.722 0.493 0.669 0.669 0.669 0.493 0.722 0.722 1.000
0.722 0.473
0.165 0.722 0.112 0.148 0.148 0.217 0.217 0.217 0.217 0.217 0.148 0.722
0.237 0.319
0.217 0.473 0.101 0.114 0.114 0.148 0.148 0.148 0.148 0.148 0.217 0.319
0.319 0.319
0.217 0.473 0.101 0.101 0.101 0.148 0.148 0.148 0.148 0.148 0.217 0.217
0.319 0.319
0.165 0.722 0.112 0.148 0.148 0.217 0.217 0.217 0.217 0.217 0.237 0.319
0.319 0.342
0.722 1.000 0.722 0.722 0.493 0.669 0.669 0.669 0.669 0.722 1.000 1.000
1.000 0.722
0.237 0.722 0.148 0.217 0.217 0.217 0.217 0.217 0.217 0.217 0.217 0.319
0.319 0.342

```
Layer 1 image number 4
0.319 0.319 0.217 0.217 0.217 0.217 0.217 0.217 0.217 0.217 0.217 0.473
0.319 0.319
0.342 0.319 0.217 0.217 0.217 0.217 0.217 0.217 0.217 0.217 0.165 0.722
0.237 0.319
1.000 1.000 1.000 0.722 0.669 0.669 0.669 0.669 0.493 0.722 0.722 1.000
0.722 0.473
0.319 0.319 0.217 0.217 0.217 0.217 0.217 0.217 0.148 0.148 0.112 0.722
0.148 0.237
0.319 0.237 0.237 0.165 0.148 0.148 0.148 0.148 0.101 0.101 0.101 0.473
0.217 0.217
0.319 0.217 0.217 0.148 0.148 0.148 0.148 0.148 0.101 0.101 0.101 0.370
0.217 0.217
0.342 0.319 0.217 0.217 0.217 0.217 0.217 0.217 0.148 0.148 0.101 0.493
0.148 0.217
1.000 1.000 1.000 0.722 0.669 0.669 0.669 0.669 0.473 0.722 0.473 1.000
0.473 0.319
0.319 0.319 0.217 0.217 0.217 0.217 0.217 0.217 0.148 0.148 0.101 0.493
0.148 0.217
0.319 0.237 0.237 0.165 0.148 0.148 0.148 0.148 0.101 0.101 0.101 0.370
0.217 0.217
0.319 0.217 0.217 0.148 0.148 0.148 0.148 0.148 0.101 0.101 0.101 0.473
0.217 0.217
0.342 0.319 0.217 0.217 0.217 0.217 0.217 0.217 0.148 0.148 0.114 0.722
0.148 0.237
1.000 1.000 1.000 0.722 0.669 0.669 0.669 0.669 0.493 0.722 0.722 1.000
0.722 0.473
0.319 0.319 0.217 0.217 0.217 0.217 0.217 0.217 0.217 0.217 0.148 0.722
0.237 0.319

Layer 1 image number 5
0.319 0.722 0.342 0.319 0.319 0.342 0.342 0.342 0.342 0.342 0.342 0.342
0.342 0.342
0.342 1.000 0.319 0.319 0.319 0.342 0.342 0.342 0.342 0.342 0.342 0.342
0.342 0.342
0.319 1.000 0.319 0.319 0.319 0.342 0.342 0.342 0.342 0.342 0.342 0.342
0.342 0.342
0.217 1.000 0.217 0.237 0.319 0.342 0.342 0.342 0.342 0.342 0.342 0.342
0.342 0.342
0.217 0.722 0.217 0.217 0.319 0.342 0.342 0.342 0.342 0.342 0.342 0.342
0.342 0.342
0.217 0.669 0.217 0.217 0.217 0.342 0.342 0.342 0.342 0.342 0.342 0.342
0.342 0.342
0.217 0.669 0.217 0.217 0.237 0.319 0.342 0.473 0.342 0.342 0.473 0.342
0.342 0.342
0.217 0.669 0.217 0.217 0.217 0.342 0.319 0.722 0.342 0.319 0.319 0.342
0.342 0.342
0.217 0.669 0.217 0.217 0.217 0.319 0.342 1.000 0.319 0.319 0.319 0.342
0.342 0.342
0.217 0.493 0.148 0.165 0.148 0.237 0.217 0.722 0.217 0.319 0.237 0.319
0.319 0.473
0.217 0.722 0.148 0.148 0.114 0.148 0.148 0.473 0.148 0.217 0.319 0.319
0.319 0.342
0.165 0.722 0.112 0.148 0.114 0.148 0.068 0.319 0.078 0.148 0.217 0.319
0.342 0.342
0.722 1.000 0.722 0.722 0.319 0.473 0.217 1.000 0.319 0.473 0.722 1.000
0.722 0.473
0.237 0.722 0.148 0.217 0.148 0.148 0.078 0.319 0.068 0.148 0.217 0.319
0.342 0.342
```

```
Layer 1 image number 6
0.319 0.473 0.217 0.217 0.217 0.217 0.217 0.217 0.217 0.217 0.217 0.319
0.319 0.319
0.237 0.722 0.165 0.217 0.217 0.217 0.217 0.217 0.217 0.217 0.237 0.319
0.319 0.342
0.722 1.000 0.722 0.722 0.493 0.669 0.669 0.669 0.669 0.722 1.000 1.000
1.000 0.722
0.165 0.722 0.112 0.148 0.148 0.217 0.217 0.217 0.217 0.217 0.217 0.319
0.319 0.342
0.217 0.473 0.101 0.114 0.114 0.148 0.148 0.148 0.148 0.148 0.217 0.217
0.319 0.319
0.217 0.473 0.101 0.101 0.101 0.148 0.148 0.148 0.148 0.148 0.217 0.319
0.319 0.319
0.165 0.722 0.112 0.148 0.148 0.217 0.217 0.217 0.217 0.217 0.165 0.722
0.237 0.319
0.722 1.000 0.722 0.722 0.493 0.669 0.669 0.669 0.493 0.722 0.722 1.000
0.722 0.473
0.237 0.722 0.148 0.217 0.217 0.217 0.217 0.217 0.148 0.148 0.112 0.722
0.148 0.237
0.319 0.319 0.237 0.165 0.148 0.148 0.148 0.148 0.101 0.101 0.101 0.473
0.217 0.217
0.319 0.217 0.217 0.148 0.148 0.148 0.148 0.148 0.101 0.101 0.101 0.473
0.217 0.217
0.342 0.319 0.217 0.217 0.217 0.217 0.217 0.217 0.148 0.148 0.114 0.722
0.148 0.237
1.000 1.000 1.000 0.722 0.669 0.669 0.669 0.669 0.493 0.722 0.722 1.000
0.722 0.473
0.319 0.319 0.217 0.217 0.217 0.217 0.217 0.217 0.217 0.217 0.148 0.722
0.237 0.319

Layer 1 image number 7
0.319 0.473 0.217 0.217 0.217 0.217 0.217 0.217 0.217 0.217 0.217 0.319
0.319 0.319
0.237 0.722 0.165 0.217 0.217 0.217 0.217 0.217 0.217 0.217 0.237 0.319
0.319 0.342
0.722 1.000 0.722 0.722 0.493 0.669 0.669 0.669 0.669 0.722 1.000 1.000
1.000 0.722
0.165 0.722 0.112 0.148 0.148 0.217 0.217 0.217 0.217 0.217 0.217 0.319
0.319 0.342
0.217 0.473 0.101 0.114 0.114 0.148 0.148 0.148 0.148 0.148 0.217 0.217
0.319 0.319
0.217 0.370 0.101 0.101 0.101 0.148 0.148 0.148 0.148 0.148 0.217 0.319
0.319 0.319
0.148 0.493 0.101 0.148 0.148 0.217 0.217 0.217 0.217 0.217 0.165 0.722
0.237 0.319
0.473 1.000 0.473 0.722 0.473 0.669 0.669 0.669 0.493 0.722 0.722 1.000
0.722 0.473
0.165 0.493 0.101 0.148 0.148 0.217 0.217 0.217 0.148 0.148 0.112 0.722
0.148 0.237
0.217 0.370 0.101 0.114 0.101 0.148 0.148 0.148 0.101 0.101 0.101 0.473
0.217 0.217
0.217 0.473 0.101 0.101 0.101 0.148 0.148 0.148 0.101 0.101 0.101 0.473
0.217 0.217
0.165 0.722 0.112 0.148 0.148 0.217 0.217 0.217 0.148 0.148 0.114 0.722
0.148 0.237
0.722 1.000 0.722 0.722 0.493 0.669 0.669 0.669 0.493 0.722 0.722 1.000
0.722 0.473
0.237 0.722 0.148 0.217 0.217 0.217 0.217 0.217 0.217 0.217 0.148 0.722
0.237 0.319
```

```
Layer 1 image number 8
0.319 0.319 0.217 0.217 0.217 0.217 0.217 0.217 0.217 0.217 0.217 0.473
0.319 0.319
0.342 0.319 0.217 0.217 0.217 0.217 0.217 0.217 0.217 0.217 0.165 0.722
0.237 0.319
1.000 1.000 1.000 0.722 0.669 0.669 0.669 0.669 0.493 0.722 0.722 1.000
0.722 0.473
0.319 0.319 0.217 0.217 0.217 0.217 0.217 0.217 0.148 0.148 0.112 0.722
0.148 0.237
0.319 0.319 0.237 0.217 0.217 0.217 0.217 0.217 0.165 0.148 0.148 0.722
0.217 0.217
0.319 0.319 0.319 0.319 0.217 0.217 0.217 0.217 0.165 0.148 0.148 0.493
0.217 0.217
0.342 0.342 0.342 0.342 0.342 0.342 0.342 0.342 0.237 0.217 0.217 0.669
0.217 0.217
0.342 0.342 0.342 0.342 0.342 0.342 0.342 0.342 0.237 0.217 0.217 0.669
0.217 0.217
0.342 0.342 0.342 0.342 0.342 0.342 0.342 0.342 0.237 0.217 0.217 0.669
0.217 0.217
0.342 0.342 0.342 0.342 0.342 0.342 0.342 0.342 0.319 0.237 0.217 0.722
0.217 0.217
0.342 0.342 0.342 0.342 0.342 0.342 0.342 0.342 0.319 0.217 0.237 1.000
0.217 0.237
0.342 0.342 0.342 0.342 0.342 0.342 0.342 0.342 0.319 0.319 0.319 1.000
0.319 0.319
0.342 0.342 0.342 0.342 0.342 0.342 0.342 0.342 0.319 0.319 0.319 1.000
0.319 0.319

Layer 1 image number 9
0.319 0.473 0.217 0.217 0.217 0.217 0.217 0.217 0.217 0.217 0.217 0.473
0.319 0.319
0.237 0.722 0.165 0.217 0.217 0.217 0.217 0.217 0.217 0.217 0.165 0.722
0.237 0.319
0.722 1.000 0.722 0.722 0.493 0.669 0.669 0.669 0.493 0.722 0.722 1.000
0.722 0.473
0.165 0.722 0.112 0.148 0.148 0.217 0.217 0.217 0.148 0.148 0.112 0.722
0.148 0.237
0.217 0.473 0.101 0.114 0.114 0.148 0.148 0.148 0.101 0.101 0.101 0.473
0.217 0.217
0.217 0.370 0.101 0.101 0.101 0.148 0.148 0.148 0.101 0.101 0.101 0.370
0.217 0.217
0.148 0.493 0.101 0.148 0.148 0.217 0.217 0.217 0.148 0.148 0.101 0.493
0.148 0.217
0.473 1.000 0.473 0.722 0.473 0.669 0.669 0.669 0.473 0.722 0.473 1.000
0.473 0.319
0.165 0.493 0.101 0.148 0.148 0.217 0.217 0.217 0.148 0.148 0.101 0.493
0.148 0.217
0.217 0.370 0.101 0.114 0.101 0.148 0.148 0.148 0.101 0.101 0.101 0.370
0.217 0.217
0.217 0.473 0.101 0.101 0.101 0.148 0.148 0.148 0.101 0.101 0.101 0.473
0.217 0.217
0.165 0.722 0.112 0.148 0.148 0.217 0.217 0.217 0.148 0.148 0.114 0.722
0.148 0.237
0.722 1.000 0.722 0.722 0.493 0.669 0.669 0.669 0.493 0.722 0.722 1.000
0.722 0.473
0.237 0.722 0.148 0.217 0.217 0.217 0.217 0.217 0.217 0.217 0.148 0.722
0.237 0.319
```

```
Layer 1 image number 10
0.319 0.473 0.217 0.217 0.217 0.217 0.217 0.217 0.217 0.217 0.217 0.473
0.319 0.319
0.237 0.722 0.165 0.217 0.217 0.217 0.217 0.217 0.217 0.217 0.165 0.722
0.237 0.319
0.722 1.000 0.722 0.722 0.493 0.669 0.669 0.669 0.493 0.722 0.722 1.000
0.722 0.473
0.165 0.722 0.112 0.148 0.148 0.217 0.217 0.217 0.148 0.148 0.112 0.722
0.148 0.237
0.217 0.473 0.101 0.114 0.114 0.148 0.148 0.148 0.101 0.101 0.101 0.473
0.217 0.217
0.217 0.473 0.101 0.101 0.101 0.148 0.148 0.148 0.101 0.101 0.101 0.370
0.217 0.217
0.165 0.722 0.112 0.148 0.148 0.217 0.217 0.217 0.148 0.148 0.101 0.493
0.148 0.217
0.722 1.000 0.722 0.722 0.493 0.669 0.669 0.669 0.473 0.722 0.473 1.000
0.473 0.319
0.237 0.722 0.148 0.217 0.217 0.217 0.217 0.217 0.148 0.148 0.101 0.493
0.148 0.217
0.319 0.473 0.237 0.217 0.217 0.217 0.217 0.217 0.148 0.148 0.148 0.493
0.217 0.217
0.319 0.319 0.319 0.319 0.217 0.217 0.217 0.217 0.217 0.165 0.148 0.473
0.217 0.217
0.342 0.342 0.342 0.342 0.342 0.342 0.342 0.342 0.319 0.217 0.237 1.000
0.217 0.237
0.342 0.342 0.342 0.342 0.342 0.342 0.342 0.342 0.319 0.319 0.319 1.000
0.319 0.319
0.342 0.342 0.342 0.342 0.342 0.342 0.342 0.342 0.319 0.319 0.319 1.000
0.319 0.319
```

Results A6.8 Maximum outputs in primitive layer

The maximum output values for the size layer are shown in Results A6.9.

```
Size layer 1 image number 1
0.658 0.047 0.060 0.085 0.147 0.574 0.574 0.147 0.085 0.060 0.658 0.038
0.000 0.068 0.092 0.142 0.312 1.000 0.312 0.142 0.092 0.068 0.000 0.602
0.068 0.063 0.063 0.086 0.149 0.583 0.583 0.149 0.086 0.062 0.068 0.060
0.092 0.089 0.090 0.090 0.153 0.609 0.609 0.153 0.089 0.088 0.092 0.086
0.142 0.153 0.155 0.156 0.157 0.630 0.630 0.156 0.153 0.153 0.142 0.148
0.312 0.613 0.623 0.631 0.639 0.642 0.642 0.625 0.610 0.607 0.312 0.576
1.000 0.613 0.623 0.631 0.639 0.642 0.642 0.625 0.610 0.607 1.000 0.576
0.312 0.153 0.155 0.156 0.158 0.642 0.642 0.158 0.153 0.153 0.312 0.148
0.142 0.089 0.090 0.090 0.155 0.623 0.623 0.155 0.090 0.088 0.142 0.086
0.092 0.063 0.063 0.088 0.152 0.604 0.604 0.152 0.088 0.062 0.092 0.060
0.068 0.048 0.063 0.089 0.153 0.613 0.613 0.153 0.089 0.063 0.068 0.047
0.000 0.068 0.092 0.142 0.312 1.000 0.312 0.142 0.092 0.068 0.000 0.645
Size layer 1 image number 2
0.113 0.244 0.620 0.244 0.114 0.000 0.087 0.150 0.590 0.590 0.150 0.087
0.112 0.241 0.608 0.248 0.115 0.000 0.086 0.148 0.578 0.578 0.148 0.086
0.113 0.244 0.620 0.248 0.115 0.000 0.087 0.150 0.590 0.590 0.150 0.087
0.113 0.112 0.091 0.092 0.153 0.613 0.613 0.153 0.090 0.110 0.110 0.086
0.243 0.239 0.156 0.159 0.154 0.598 0.598 0.159 0.155 0.235 0.235 0.148
0.617 0.604 0.632 0.648 0.613 0.598 0.600 0.647 0.625 0.588 0.587 0.581
0.243 0.239 0.632 0.648 0.613 1.000 0.600 0.647 0.625 0.235 0.235 0.581
0.113 0.112 0.156 0.159 0.154 1.000 0.598 0.159 0.155 0.110 0.110 0.148
0.074 0.073 0.091 0.092 0.154 0.619 0.619 0.154 0.090 0.072 0.072 0.086
0.055 0.056 0.064 0.089 0.154 0.614 0.614 0.154 0.089 0.063 0.053 0.061
0.044 0.047 0.061 0.087 0.150 0.588 0.588 0.150 0.087 0.061 0.047 0.047
0.000 0.000 0.302 1.000 0.302 0.000 0.260 0.692 0.260 0.000 0.000 0.592
```

Size layer 1 image number 3
0.641 0.640 0.062 0.088 0.151 0.600 0.600 0.151 0.088 0.062 0.658 0.042
0.000 0.000 0.078 0.111 0.194 1.000 1.000 0.194 0.111 0.078 0.000 0.602
0.150 0.048 0.063 0.089 0.153 0.609 0.609 0.153 0.089 0.073 0.194 0.072
0.589 0.061 0.089 0.091 0.156 0.633 0.633 0.156 0.091 0.112 1.000 0.109
0.589 0.087 0.154 0.156 0.157 0.637 0.637 0.157 0.155 0.241 1.000 0.234
0.150 0.150 0.618 0.632 0.639 0.639 0.639 0.632 0.620 0.609 0.194 0.585
0.000 0.593 0.618 0.632 0.639 1.000 0.639 0.632 0.620 0.241 0.000 0.645
0.194 0.593 0.154 0.156 0.157 0.571 0.571 0.156 0.155 0.112 0.153 0.109
1.000 0.150 0.089 0.091 0.153 0.609 0.609 0.153 0.090 0.073 0.607 0.072
1.000 0.087 0.064 0.090 0.156 0.630 0.630 0.156 0.090 0.064 0.607 0.053
0.194 0.061 0.061 0.087 0.150 0.588 0.588 0.150 0.087 0.061 0.153 0.042
0.000 0.078 0.111 0.194 1.000 1.000 0.194 0.111 0.078 0.000 0.000 0.000

Size layer 1 image number 4
0.641 0.640 0.062 0.088 0.151 0.600 0.600 0.151 0.088 0.062 0.658 0.039
0.000 0.000 0.078 0.111 0.194 1.000 1.000 0.194 0.111 0.078 0.000 0.602
0.151 0.153 0.063 0.089 0.153 0.609 0.609 0.153 0.089 0.062 0.194 0.060
0.599 0.608 0.090 0.091 0.156 0.633 0.633 0.156 0.091 0.088 1.000 0.086
0.599 0.608 0.155 0.157 0.157 0.635 0.635 0.157 0.153 0.153 1.000 0.148
0.151 0.153 0.625 0.639 0.639 0.639 0.639 0.625 0.610 0.607 0.194 0.576
0.000 0.000 0.625 0.639 0.639 1.000 1.000 0.625 0.610 0.607 0.000 0.587
0.151 0.153 0.155 0.157 0.157 0.615 0.615 0.155 0.153 0.153 0.194 0.148
0.599 0.608 0.090 0.091 0.157 0.635 0.635 0.157 0.091 0.088 1.000 0.086
0.599 0.608 0.064 0.091 0.158 0.642 0.642 0.158 0.091 0.064 1.000 0.060
0.151 0.153 0.063 0.089 0.153 0.611 0.611 0.153 0.089 0.063 0.194 0.047
0.000 0.000 0.078 0.111 0.194 1.000 1.000 0.194 0.111 0.078 0.000 0.645

Size layer 1 image number 5
0.000 0.042 0.053 0.072 0.110 0.236 0.590 0.236 0.110 0.072 0.053 0.042
0.000 0.047 0.053 0.071 0.109 0.233 0.578 0.233 0.109 0.071 0.053 0.047
0.000 0.061 0.065 0.072 0.110 0.236 0.590 0.236 0.110 0.072 0.072 0.061
0.078 0.087 0.092 0.091 0.151 0.599 0.599 0.151 0.088 0.088 0.110 0.086
0.111 0.149 0.158 0.156 0.156 0.632 0.632 0.156 0.152 0.151 0.235 0.148
0.194 0.585 0.646 0.633 0.620 0.639 0.639 0.603 0.606 0.597 0.587 0.581
1.000 0.585 0.646 0.633 0.620 0.632 0.632 0.593 0.610 0.603 0.235 0.581
1.000 0.155 0.626 0.626 0.155 0.151 0.000 0.150 0.236 0.590 0.236 0.148
0.194 0.087 0.092 0.091 0.156 0.632 0.632 0.156 0.091 0.088 0.072 0.086
0.111 0.061 0.065 0.090 0.155 0.620 1.000 0.155 0.090 0.063 0.053 0.061
0.078 0.047 0.061 0.087 0.150 0.588 0.588 0.150 0.087 0.061 0.047 0.047
0.000 0.142 0.312 1.000 0.312 0.142 0.000 0.260 0.693 0.260 0.000 0.592

Size layer 1 image number 6
0.658 0.046 0.060 0.084 0.145 0.563 0.563 0.145 0.084 0.617 0.616 0.592
0.000 0.070 0.100 0.174 0.778 0.778 0.174 0.100 0.070 0.000 0.000 0.000
0.194 0.075 0.063 0.084 0.145 0.560 0.560 0.145 0.084 0.059 0.150 0.048
1.000 0.115 0.090 0.091 0.152 0.600 0.600 0.152 0.089 0.062 0.589 0.062
1.000 0.247 0.155 0.157 0.157 0.617 0.617 0.156 0.154 0.087 0.589 0.087
0.194 0.632 0.622 0.638 0.639 0.639 0.639 0.632 0.615 0.150 0.150 0.151
0.000 0.247 0.622 0.638 0.639 1.000 0.639 0.632 0.615 0.592 0.000 0.602
0.150 0.115 0.155 0.157 0.157 0.617 0.617 0.156 0.154 0.592 0.194 0.594
0.589 0.075 0.090 0.091 0.155 0.623 0.623 0.155 0.090 0.150 1.000 0.151
0.589 0.056 0.064 0.091 0.158 0.642 0.642 0.158 0.091 0.087 1.000 0.087
0.150 0.048 0.063 0.089 0.153 0.611 0.611 0.153 0.089 0.063 0.194 0.062
0.000 0.000 0.078 0.111 0.194 1.000 1.000 0.194 0.111 0.078 0.000 0.645

Size layer 1 image number 7
0.658 0.046 0.060 0.084 0.145 0.563 0.563 0.145 0.084 0.617 0.616 0.592
0.000 0.070 0.100 0.174 0.778 0.778 0.174 0.100 0.070 0.000 0.000 0.000
0.194 0.063 0.063 0.084 0.145 0.560 0.560 0.145 0.084 0.059 0.150 0.048
1.000 0.089 0.090 0.090 0.152 0.600 0.600 0.152 0.089 0.062 0.589 0.062
1.000 0.153 0.155 0.156 0.157 0.608 0.608 0.156 0.154 0.087 0.589 0.087
0.194 0.613 0.623 0.631 0.639 0.639 0.639 0.632 0.615 0.150 0.150 0.151
0.000 0.613 0.623 0.631 0.639 1.000 0.639 0.632 0.615 0.592 0.000 0.602
0.194 0.153 0.155 0.156 0.157 0.602 0.602 0.156 0.154 0.592 0.194 0.594
1.000 0.089 0.090 0.090 0.153 0.613 0.613 0.153 0.089 0.150 1.000 0.151
1.000 0.063 0.063 0.089 0.154 0.613 0.613 0.154 0.089 0.087 1.000 0.087
0.194 0.048 0.063 0.089 0.153 0.613 0.613 0.153 0.089 0.063 0.194 0.062
0.000 0.068 0.092 0.142 0.312 1.000 0.312 0.142 0.092 0.068 0.000 0.645

```
Size layer 1 image number 8
0.641 0.640 0.062 0.088 0.151 0.600 0.600 0.151 0.088 0.062 0.658 0.039
0.000 0.000 0.078 0.111 0.194 1.000 1.000 0.194 0.111 0.078 0.000 0.602
0.047 0.048 0.062 0.089 0.153 0.609 0.609 0.153 0.089 0.063 0.078 0.061
0.061 0.061 0.088 0.091 0.157 0.639 0.639 0.157 0.091 0.089 0.111 0.087
0.087 0.087 0.151 0.156 0.158 0.646 0.646 0.158 0.158 0.153 0.194 0.150
0.150 0.149 0.599 0.632 0.632 0.635 0.635 0.642 0.641 0.610 1.000 0.591
0.591 0.587 0.599 0.632 0.632 0.635 0.635 0.642 0.641 0.610 1.000 0.591
0.591 0.587 0.151 0.156 0.157 0.635 0.635 0.158 0.158 0.153 0.194 0.150
0.150 0.149 0.088 0.091 0.157 0.635 0.635 0.157 0.091 0.089 0.111 0.087
0.087 0.087 0.063 0.090 0.155 0.626 0.626 0.155 0.090 0.063 0.078 0.061
0.061 0.064 0.090 0.156 0.628 0.628 0.156 0.090 0.064 0.049 0.000 0.640
0.050 0.064 0.091 0.158 0.642 0.642 0.158 0.091 0.064 0.050 0.000 0.641

Size layer 1 image number 9
0.658 0.047 0.060 0.085 0.147 0.574 0.574 0.147 0.085 0.060 0.658 0.038
0.000 0.068 0.092 0.142 0.312 1.000 0.312 0.142 0.092 0.068 0.000 0.602
0.194 0.063 0.063 0.086 0.149 0.583 0.583 0.149 0.086 0.062 0.194 0.060
1.000 0.089 0.090 0.090 0.153 0.611 0.611 0.153 0.089 0.088 1.000 0.086
1.000 0.153 0.155 0.156 0.157 0.600 0.600 0.155 0.153 0.153 1.000 0.148
0.194 0.613 0.623 0.631 0.639 0.639 0.639 0.625 0.610 0.607 0.194 0.576
0.000 0.613 0.623 0.631 0.639 0.766 0.639 0.625 0.610 0.607 0.000 0.587
0.194 0.153 0.155 0.156 0.157 0.594 0.594 0.155 0.153 0.153 0.194 0.148
1.000 0.089 0.090 0.090 0.151 0.600 0.600 0.151 0.089 0.088 1.000 0.086
1.000 0.063 0.063 0.089 0.154 0.613 0.613 0.154 0.089 0.063 1.000 0.060
0.194 0.048 0.063 0.089 0.153 0.613 0.613 0.153 0.089 0.063 0.194 0.047
0.000 0.068 0.092 0.142 0.312 1.000 0.312 0.142 0.092 0.068 0.000 0.645

Size layer 1 image number 10
0.658 0.047 0.060 0.085 0.147 0.574 0.574 0.147 0.085 0.060 0.658 0.039
0.000 0.068 0.092 0.142 0.312 1.000 0.312 0.142 0.092 0.068 0.000 0.602
0.194 0.062 0.062 0.086 0.149 0.583 0.583 0.149 0.086 0.063 0.194 0.061
1.000 0.089 0.087 0.089 0.153 0.611 0.611 0.153 0.091 0.089 1.000 0.087
1.000 0.153 0.151 0.154 0.156 0.611 0.611 0.158 0.158 0.153 1.000 0.150
0.194 0.609 0.594 0.614 0.632 0.632 0.632 0.645 0.641 0.610 0.194 0.591
0.000 0.609 0.594 0.614 0.632 1.000 0.632 0.645 0.641 0.610 0.000 0.591
0.109 0.153 0.151 0.154 0.156 0.608 0.608 0.158 0.158 0.153 0.297 0.150
0.233 0.089 0.087 0.091 0.157 0.636 0.636 0.157 0.091 0.089 1.000 0.087
0.580 0.062 0.065 0.091 0.158 0.643 0.643 0.158 0.091 0.064 0.297 0.061
0.233 0.064 0.090 0.156 0.628 0.628 0.156 0.090 0.064 0.049 0.000 0.640
0.109 0.064 0.091 0.158 0.642 0.642 0.158 0.091 0.064 0.050 0.000 0.641
```

Results A6.9 Maximum outputs in size layer

The maximum output values for the corner layer are shown in Results A6.10.

```
Corn layer 1 image number 1
0.283 0.379 0.392 0.431 0.502 0.345 0.292 0.283 0.281 0.146
0.297 0.377 0.386 0.415 0.568 0.488 0.393 0.377 0.373 0.275
0.300 0.381 0.390 0.419 0.574 0.493 0.397 0.381 0.377 0.278
0.306 0.398 0.410 0.448 0.634 0.692 0.551 0.530 0.524 0.346
0.327 0.492 0.507 0.555 0.845 1.000 0.692 0.662 0.654 0.403
0.365 0.471 0.482 0.517 0.826 0.609 0.490 0.471 0.467 0.512
0.327 0.418 0.427 0.459 0.629 0.539 0.435 0.418 0.414 0.421
0.306 0.389 0.398 0.428 0.586 0.503 0.406 0.389 0.385 0.385
0.300 0.381 0.390 0.419 0.574 0.493 0.397 0.381 0.377 0.378
0.228 0.300 0.306 0.328 0.432 0.328 0.306 0.300 0.297 0.283
```

Corn layer 1 image number 2
0.064 0.133 0.239 0.220 0.063 0.245 0.244 0.219 0.243 0.154
0.064 0.134 0.242 0.222 0.064 0.247 0.247 0.221 0.244 0.155
0.063 0.118 0.205 0.183 0.071 0.207 0.207 0.183 0.207 0.132
0.066 0.124 0.214 0.193 0.074 0.217 0.217 0.192 0.217 0.138
0.066 0.141 0.253 0.221 0.080 0.246 0.244 0.218 0.217 0.138
0.341 0.510 1.000 0.704 0.081 0.871 0.875 0.681 0.618 0.460
0.346 0.510 1.000 0.704 0.211 0.871 0.875 0.681 0.627 0.466
0.052 0.116 0.202 0.181 0.204 0.206 0.204 0.180 0.168 0.108
0.052 0.116 0.200 0.184 0.214 0.250 0.246 0.221 0.207 0.132
0.059 0.135 0.242 0.210 0.187 0.220 0.244 0.219 0.197 0.126

Corn layer 1 image number 3
0.188 0.264 0.271 0.319 0.312 0.342 0.292 0.285 0.331 0.145
0.111 0.174 0.180 0.207 0.254 0.254 0.180 0.176 0.211 0.063
0.355 0.457 0.473 0.603 0.927 0.927 0.476 0.461 0.584 0.072
0.355 0.457 0.473 0.603 0.927 0.927 0.476 0.461 0.584 0.335
0.239 0.317 0.400 0.460 0.567 0.595 0.501 0.488 0.511 0.575
0.285 0.497 0.522 0.623 0.961 0.419 0.363 0.355 0.347 0.401
0.315 0.494 0.512 0.593 0.898 0.765 0.514 0.498 0.490 0.609
0.367 0.475 0.492 0.618 1.000 0.728 0.494 0.479 0.471 0.584
0.367 0.475 0.491 0.617 1.000 0.728 0.489 0.475 0.467 0.483
0.241 0.318 0.328 0.375 0.535 0.453 0.318 0.313 0.298 0.298

Corn layer 1 image number 4
0.193 0.260 0.264 0.272 0.313 0.266 0.291 0.283 0.280 0.145
0.133 0.174 0.176 0.181 0.256 0.256 0.181 0.176 0.174 0.064
0.455 0.471 0.477 0.494 1.000 1.000 0.494 0.477 0.471 0.073
0.455 0.471 0.477 0.494 1.000 1.000 0.494 0.477 0.471 0.152
0.207 0.240 0.294 0.308 0.409 0.378 0.368 0.359 0.355 0.179
0.193 0.325 0.329 0.337 0.411 0.397 0.360 0.351 0.347 0.173
0.133 0.174 0.177 0.184 0.260 0.257 0.222 0.217 0.214 0.075
0.431 0.443 0.450 0.466 1.000 1.000 0.466 0.450 0.443 0.073
0.361 0.443 0.450 0.466 1.000 1.000 0.466 0.450 0.443 0.295
0.164 0.240 0.243 0.251 0.381 0.381 0.251 0.243 0.240 0.229

Corn layer 1 image number 5
0.291 0.412 0.462 0.418 0.494 0.296 0.542 0.636 0.542 0.411
0.291 0.415 0.466 0.421 0.498 0.298 0.581 0.686 0.582 0.436
0.307 0.422 0.474 0.425 0.494 0.321 0.420 0.491 0.421 0.328
0.311 0.435 0.490 0.439 0.510 0.329 0.435 0.507 0.435 0.338
0.322 0.536 0.604 0.541 0.630 0.383 0.535 0.627 0.535 0.406
0.372 0.511 0.582 0.511 0.605 0.384 0.506 0.598 0.506 0.389
0.380 0.524 0.596 0.524 0.620 0.380 0.518 0.613 0.518 0.372
0.322 0.437 0.487 0.437 0.514 0.331 0.432 0.504 0.432 0.313
0.470 0.681 1.000 0.680 0.844 0.493 0.676 0.853 0.677 0.459
0.236 0.328 0.362 0.328 0.307 0.284 0.325 0.374 0.325 0.219

Corn layer 1 image number 6
0.355 0.379 0.389 0.418 0.467 0.587 0.510 0.492 0.488 0.329
0.388 0.395 0.401 0.416 0.668 0.479 0.416 0.401 0.397 0.287
0.452 0.468 0.475 0.492 1.000 0.587 0.492 0.475 0.470 0.327
0.452 0.468 0.475 0.492 1.000 0.587 0.492 0.475 0.470 0.327
0.258 0.379 0.384 0.396 0.588 0.446 0.396 0.384 0.381 0.239
0.224 0.320 0.324 0.332 0.365 0.394 0.356 0.347 0.345 0.171
0.124 0.194 0.196 0.202 0.289 0.235 0.202 0.196 0.195 0.069
0.353 0.451 0.458 0.474 1.000 0.589 0.474 0.457 0.453 0.072
0.351 0.451 0.457 0.474 1.000 0.589 0.474 0.457 0.453 0.290
0.164 0.238 0.241 0.249 0.382 0.281 0.249 0.241 0.239 0.227

```
Corn layer 1 image number 7
0.283 0.380 0.395 0.439 0.468 0.626 0.513 0.489 0.486 0.329
0.312 0.397 0.406 0.437 0.630 0.509 0.417 0.397 0.394 0.287
0.368 0.476 0.486 0.522 0.823 0.609 0.499 0.476 0.473 0.329
0.368 0.476 0.486 0.522 0.823 0.609 0.499 0.476 0.473 0.329
0.312 0.492 0.506 0.554 0.759 1.000 0.690 0.654 0.649 0.410
0.283 0.494 0.510 0.563 0.668 0.418 0.364 0.352 0.350 0.174
0.312 0.397 0.406 0.437 0.769 0.509 0.418 0.397 0.395 0.287
0.365 0.473 0.483 0.518 1.000 0.605 0.496 0.473 0.470 0.328
0.365 0.473 0.483 0.518 1.000 0.610 0.492 0.473 0.470 0.487
0.239 0.315 0.321 0.344 0.540 0.344 0.321 0.315 0.313 0.297

Corn layer 1 image number 8
0.064 0.059 0.060 0.069 0.071 0.064 0.062 0.062 0.062 0.145
0.000 0.000 0.000 0.000 0.246 0.246 0.000 0.000 0.000 0.073
0.000 0.000 0.000 0.000 0.248 0.248 0.000 0.000 0.000 0.061
0.000 0.000 0.000 0.000 0.254 0.254 0.000 0.000 0.000 0.062
0.232 0.242 0.245 0.251 1.000 1.000 0.251 0.245 0.242 0.069
0.232 0.242 0.245 0.251 1.000 1.000 0.251 0.245 0.242 0.070
0.000 0.000 0.000 0.000 0.271 0.271 0.000 0.000 0.000 0.070
0.000 0.000 0.000 0.000 0.248 0.248 0.000 0.000 0.000 0.067
0.000 0.000 0.000 0.000 0.246 0.246 0.000 0.000 0.000 0.066
0.000 0.000 0.000 0.000 0.220 0.220 0.000 0.000 0.000 0.061

Corn layer 1 image number 9
0.283 0.379 0.392 0.431 0.502 0.342 0.295 0.283 0.281 0.147
0.312 0.397 0.406 0.437 0.598 0.509 0.418 0.397 0.393 0.288
0.368 0.476 0.486 0.522 0.819 0.625 0.500 0.476 0.471 0.331
0.368 0.476 0.486 0.522 0.819 0.625 0.500 0.476 0.471 0.330
0.312 0.492 0.506 0.554 0.839 1.000 0.691 0.654 0.647 0.411
0.283 0.494 0.510 0.563 0.941 0.976 0.687 0.650 0.643 0.401
0.312 0.397 0.406 0.437 0.598 0.509 0.418 0.397 0.393 0.308
0.365 0.473 0.483 0.518 0.801 0.605 0.496 0.473 0.468 0.335
0.365 0.473 0.483 0.518 0.801 0.610 0.492 0.473 0.468 0.486
0.239 0.315 0.321 0.344 0.451 0.344 0.321 0.315 0.312 0.297

Corn layer 1 image number 10
0.175 0.184 0.187 0.207 0.524 0.064 0.063 0.062 0.062 0.146
0.191 0.193 0.196 0.210 0.615 0.230 0.197 0.193 0.191 0.288
0.234 0.236 0.239 0.251 1.000 0.276 0.241 0.236 0.234 0.328
0.234 0.236 0.239 0.251 1.000 0.276 0.241 0.236 0.234 0.334
0.123 0.144 0.146 0.154 0.546 0.167 0.147 0.144 0.143 0.239
0.070 0.070 0.072 0.081 0.393 0.075 0.074 0.073 0.073 0.170
0.000 0.000 0.000 0.000 0.265 0.000 0.000 0.000 0.000 0.067
0.232 0.234 0.237 0.249 1.000 0.274 0.239 0.234 0.232 0.069
0.000 0.000 0.000 0.000 0.272 0.000 0.000 0.000 0.000 0.070
0.000 0.000 0.000 0.000 0.268 0.000 0.000 0.000 0.000 0.069
```

Results A6.10 Maximum outputs in corner layer

The maximum output values for the size 2 layer are shown in Results A6.11.

```
Size layer 1 image number 1
0.072 0.151 0.151 0.154 0.148 0.148 0.071 0.065
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.141
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.141
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.155
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.144
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.144
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.069
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.045
```

Size layer 1 image number 2
0.000 0.000 0.045 0.541 0.042 0.000 0.000 0.000
0.000 0.000 0.068 0.537 0.063 0.000 0.000 0.000
0.000 0.000 0.142 0.542 0.131 0.000 0.000 0.000
0.000 0.000 0.068 0.558 0.063 0.000 0.000 0.000
0.000 0.000 0.045 0.795 0.042 0.000 0.000 0.000
0.000 0.000 0.059 0.782 0.139 0.000 0.000 0.000
0.000 0.000 0.122 0.535 0.139 0.000 0.000 0.000
0.000 0.000 0.122 0.537 0.138 0.000 0.000 0.000

Size layer 1 image number 3
0.069 0.146 0.146 0.147 0.070 0.146 0.146 0.149
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.149
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.136
0.049 0.076 0.157 0.157 0.049 0.074 0.155 0.142
0.541 0.544 0.569 1.000 0.819 0.549 0.550 0.541
0.069 0.145 0.145 0.148 0.045 0.143 0.143 0.045
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000

Size layer 1 image number 4
0.069 0.146 0.146 0.146 0.146 0.146 0.146 0.147
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.149
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.033
0.072 0.153 0.154 0.156 0.152 0.152 0.073 0.067
0.523 0.523 0.525 1.000 1.000 0.525 0.523 0.523
0.068 0.145 0.146 0.146 0.146 0.146 0.070 0.067
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.033
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.025

Size layer 1 image number 5
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.152 0.000 0.000 0.000
0.000 0.000 0.000 0.075 0.340 0.026 0.000 0.000
0.000 0.000 0.000 0.047 0.770 0.023 0.000 0.000

Size layer 1 image number 6
0.072 0.153 0.153 0.153 0.150 0.150 0.072 0.045
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.073 0.153 0.153 0.153 0.151 0.151 0.072 0.045
0.632 0.635 0.651 1.000 0.157 0.153 0.152 0.153
0.069 0.146 0.146 0.141 0.147 0.147 0.071 0.045
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.024
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.019

Size layer 1 image number 7
0.073 0.153 0.153 0.154 0.151 0.150 0.072 0.045
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.050 0.075 0.156 0.155 0.150 0.146 0.147 0.067
0.251 0.251 0.256 1.000 0.247 0.619 0.243 0.246
0.071 0.148 0.148 0.155 0.148 0.147 0.071 0.156
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.075
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.045

Size layer 1 image number 8
0.062 0.127 0.127 0.127 0.041 0.063 0.131 0.063
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.126
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.061
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.040
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.114
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.114
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.114
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.055

```
Size layer 1 image number 9
0.072 0.150 0.150 0.155 0.148 0.148 0.071 0.065
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.142
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.141
0.051 0.076 0.158 0.157 0.149 0.148 0.148 0.141
0.252 0.252 0.253 0.310 0.246 0.621 0.244 0.244
0.072 0.150 0.150 0.155 0.148 0.148 0.071 0.155
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.074
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.045

Size layer 1 image number 10
0.069 0.143 0.143 0.033 0.045 0.141 0.141 0.136
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.143
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.042
0.070 0.145 0.145 0.033 0.046 0.147 0.147 0.064
0.499 0.501 0.509 1.000 0.505 0.499 0.498 0.498
0.061 0.127 0.127 0.030 0.042 0.133 0.133 0.064
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.039
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.029
```

Results A6.11 Maximum outputs in size 2 layer

The network was unable to correctly classify noisy images.

A6.2 NEOCOGNITRON

A four layer Neocognitron was constructed with the network architecture shown in Table A6.2.

Layer Type	Planes	Neuron Rows	Neuron Columns	Synapse Rows	Synapse Columns
Simple 1	13	15	15	7	7
Complex 1	13	15	15	7	7
Simple 2	16	11	11	5	5
Complex 2	16	11	11	5	5
Simple 3	17	7	7	5	5
Complex 3	17	7	7	5	5
Simple 4	10	3	3	5	5
Complex 4	10	1	1	3	3

Table A6.2 Neocognitron architecture

The neuron weight distribution, calculation, and updating parameters are given in Table A6.3.

Layer	Selectivity	Saturation	Inhibitory Fall Off	Complex Fall Off	Updating Gain
1	1.7	0.25	0.5	0.5	1000
2	4.0	0.25	0.5	0.5	1000
3	1.5	0.25	0.5	0.5	1000
4	1.0	1.0	0.5	0.5	1000

Table A6.3 Neocognitron neuron and teaching parameters

Each training image was presented once, and Table A6.4 shows the classification performance for both the training set and the larger images.

Image	Training Set		Larger Images	
	Classified Plane	Neuron Value	Classified Plane	Neuron Value
0	1	0.467	1	0.364
1	2	0.455	2	0.444
2	3	0.480	3	0.470
3	4	0.484	4	0.477
4	5	0.478	2	0.360
5	6	0.479	6	0.471
6	7	0.480	7	0.473
7	8	0.486	8	0.438
8	9	0.473	9	0.469
9	10	0.486	10	0.478

Table A6.4 Neocognitron classification performance